

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anej Lekše

Avtomatizacija delavniškega dnevnika

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: dr. Andrej Brodnik

Ljubljana, 2020

COPYRIGHT. Rezultati diplomske naloge so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavo in koriščenje rezultatov diplomske naloge je potrebno pisno privoljenje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil L^AT_EX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogu:

Tematika naloge:

Preveri, ali so glasovni pomočniki v trenutnem stanju primerni za pomoč pri pisanju delavniških dnevnikov. Seznani se z obstoječimi rešitvami za sestavljanje delavniških dnevnikov in jih analiziraj. Po analizi področja se loti izdelave svojega sistema za sestavljanje delavniških dnevnikov, ki vključuje glasovnega pomočnika.

Zahvaljujem se mentorju dr. Andreju Brodniku za odlično usmerjanje pri delu, očetu in mami za neizmerno podporo, Roku Bermežu in podjetju Hisense Gorenje Europe za slikovno gradivo in Meti za pomoč.

Hvala vsem.

Tii. Hvala.

Kazalo

Povzetek

Abstract

1	Uvod	1
1.1	Opis področja dela	1
1.2	Struktura diplomske naloge	2
2	Pregled problema in rešitve	3
2.1	Opis problema in uporabljenih tehnologij	3
2.2	Funkcionalne zahteve	4
2.3	Uporabljene tehnologije	5
2.4	Obstoječe rešitve	8
3	Načrtovanje in razvoj sistema	15
3.1	Definicija funkcionalnosti	15
3.2	Načrt sistema za sestavljanje opisov tehnoloških postopkov . .	17
3.3	Strežnik	19
3.4	Implementacija strežnika in mobilne aplikacije	22
3.5	Uvedba glasovnega pomočnika	42
3.6	Dodelave mobilne aplikacije	52
4	Ovrednotenje funkcionalnosti	55
5	Zaključek	59

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application Programming Interface	Vmesnik za programiranje aplikacije
AWS	Amazon Web Services	Amazonove spletne storitve
FIFO	First In First Out	Prvi noter, prvi ven
HTML	HyperText Markup Language	Označitveni jezik za hipertekst
HTTP	HyperText Transfer Protocol	Protokol za prenos hiperteksta
IP	Internet Protocol	Internetni protokol
JSON	JavaScript Object Notation	JavaScript zapis objektov
JWT	JSON Web Token	JSON spletni žeton
MIT	Massachusetts Institute of Technology	Tehnološki inštitut Massachussettsa
MVVM	Model View View-Model	Model Pogled Pogled-Model
NLU	Natural Langugage Understanding	Razumevanje naravnega govora
SQS	Simple Queue Service	Preprosta vrstna storitev
UI	User Interface	Uporabniški vmesnik
URL	Universal Resource Locator	Univerzalni lokator virov
XAML	Extensible Application Markup Langugage	Razširljiv aplikacijski označitveni jezik

Povzetek

Naslov: Avtomatizacija delavníškega dnevnika

Avtor: Anej Lekše

Diplomsko delo obravnava področje vpeljave komercialno dostopnih glasovnih pomočnikov v programske rešitve kot dodatni uporabniški vmesnik. V nalogi raziščemo, kako v računalniški sistem vpeljemo glasovnega pomočnika in vpliv glasovnega pomočnika na uporabo sistema. Cilj je implementirati in preizkusiti računalniški sistem z glasovnim pomočnikom, s pomočjo katerega lahko narekujemo zapiske med delom. Zapiske lahko naknadno urejamo s pomočjo mobilne aplikacije. Uporabili smo Amazon Alexa zaradi enostavne izdelave lastnih programov z Alexa Skills Kit ogrodjem. Ugotovili smo, da Amazon Alexa ni bila zmožna razpozнатi izgovorjenih stavkov z veliko spremenljajočimi se besedami. Uspešna je bila pri razpoznavanju stavkov, ki so se v celoti ujemali z naučenim razpoznavnim modelom. Kljub temu je čas, potreben za razpoznavo glasovnega ukaza prepočasen, da bi bil sistem, v času izdelave diplome, učinkovit.

Ključne besede: mobilni razvoj, glasovni pomočniki, razpoznavanje glasu, informacijski sistemi.

Abstract

Title: Workshop report automatization

Author: Anej Lekše

The thesis deals with the process of incorporating a commercially available voice assistant into a software solution as an additional user interface. We investigate how to implement a voice assistant in a software solution and how it effects its workflow. The goal of thesis is implementing and testing a system with a voice assistant that could be used to write workshop report steps during work. These steps can later be edited with a mobile application. We used Amazon Alexa, as it offers simple programming with Alexa Skills Kit. We found that Amazon Alexa was unsuitable for literal dictation of longer, varying text. It was successful at recognising phrases that fully or mostly matched its learned model. Regardless, Alexa's ability to process third party voice commands, at the time of this research, was too slow to be deemed efficient.

Keywords: mobile development, voice assistants, voice recognition, information systems.

Poglavlje 1

Uvod

1.1 Opis področja dela

Diplomsko delo obravnava področje vpeljave glasovnih pomočnikov v programske rešitve kot dodatni uporabniški vmesnik.

Za področje, ki ga želimo z našo rešitvijo izboljšati, smo si izbrali sestavljanje delavniškega dnevnika (v nadaljevanju imenovanega tudi opis tehnološkega postopka). Kljub temu se bo naš sistem lahko uporabil tudi za sestavljanje drugačnih vrst poročil in navodil. Med te spadajo na primer kuharski recepti, laboratorijski dnevniki itd.

Opis tehnološkega postopka ali delavniški dnevnik je dokument, ki po korakih predstavi postopek izdelave izdelka.

V sklopu diplomske naloge smo želeli izdelati specializiran sistem za sestavljanje delavniških dnevnikov in preveriti, ali lahko z vpeljavo glasovnega pomočnika delo s sistemom izboljšamo. Sistem bi sestavljal glasovni pomočnik, ki bi služil za narekovanje opomb, mobilna aplikacija, preko katere bi lahko urejali zapiske, in strežnik, ki bi hranil podatke.

Raziskati želimo, kako vpeljati glasovnega pomočnika Amazon Alexa v programsko rešitev in kako vpeljava Alexe vpliva na interakcijo uporabnika s sistemom.

1.2 Struktura diplomske naloge

Diplomsko delo pričenjamo s predstavitvijo področja našega dela in kratko opišemo problem in možno rešitev. Opredelimo problematiko našega dela.

Začnemo z raziskavo obstoječih rešitev za sestavljanje opisa tehnološkega postopka. Nato opišemo, kaj trenutne rešitve te problematike ponujajo in poskusimo opredeliti njihove prednosti ter slabosti. Lotimo se opisa tehnologij, ki smo jih pri pisanju diplome uporabili.

V naslednjem poglavju se lotimo načrtovanja specializiranega sistema z glasovnim pomočnikom za pomoč pri pisanju delavniških dnevnikov. Natančno definiramo funkcionalnosti sistema, utemeljimo odločitev za izbiro Amazon Alexa, AWS Simple Queue Service, AWS Lambda in ogrodja Xamarin.

Predstavimo še Alexa „Skill“ (v nadaljevanju Alexa veščino), strežniški program in mobilno aplikacijo. Funkcionalnosti sistema testiramo.

V zadnjem poglavju opišemo možnosti nadaljnega razvoja projekta.

Poglavlje 2

Pregled problema in rešitve

2.1 Opis problema in uporabljenih tehnologij

2.1.1 Delavniški dnevnik

Delavniški dnevnik je splošen izraz za tehniški dokument, ki po korakih predstavi potek tehnološkega postopka [12]. V podjetju Hisense Gorenje Europe takšne dokumente imenujejo opis tehnološkega postopka (slika 2.1).

Delavniški dnevnik je sestavljen iz zaporedja korakov, ki si sledijo v časovnem zaporedju. Vsak korak sestavljata opis postopka in predvideno trajanje. Koraki lahko vsebujejo tudi slikovne razlage.

Delavniški dnevnički imajo lahko definirane tudi kontrolne postopke za izdelek, orodje, ki ga potrebujemo za izdelavo, in seznam možnih nevarnosti pri delu.

Delavniški dnevnički se uporabljajo v proizvodnih obratih tovarn kot navodila in oporne točke za sestavljalce. Pogosto se uporabljajo tudi kot učni pripomoček za spremljanje napredka, npr. dijakov-vajencev na praksi [12, 8].

C	D	E	F	G	H	I	J	K
Delovni	Postopek	Kratki tekst postop.	Osnovna ko	Stand.	Enota stand.	Stand.	Stand.	Stevilo zaposlenih
OV847P 0010		Operacija planiranja	1	1 S		41.9	0	0
OV847A 0020		sestavljalec2- trojček	1	0 S		47	3	3
OV660B 0030		kontrola ohišij	1	0 S		47	1	1
OV660A 0040		čiščenje po puru	1	0 S		47	1	3
OV847A 0050		sestavljalec 2	1	0 S		45	15	15
OV847A 0060		sestavljalec 1	1	0 S		45	30	30
OV847B 0070		strojniki embalirnega stroja	1	0 S		47	1	1
OV847B 0080		pregledovalec sklopov	1	0 S		47	6	6
OV847O 0090		mehanik gospodinjskih aparatov	1	0 S		47	3	3
OV847B 0100		varilec gasflux, lak popravilo (1,5)	1	0 S		47	3	2
OV847S 0110		operacija brez delavca	1	27 S		47	0	0
OV847S 0120		vakuum krog	1	22 S		47	0	0
OV847S 0130		100% kontrola	1	19 S		47	0	0
OV847S 0140		povezava montaža - opremljanje	1	6 S		47	0	0
OV847A 0150		Vrijenje nosilca predala	1	0 S		47	1	1
OV847A 0160		Vstavljanje ventilatorja	1	0 S		47	1	1
OV847A 0170		Sestava profilov	1	0 S		47	1	1
OV847A 0180		Odstranjevanje folije	1	0 S		47	2	2
OV847P 0190		Sestava steklene police	1	0 S		19.7	1	4.1
OV847A 0200		Feniranje bulžirke	1	0 S		22.32	1	1
OV784P 0010		Linija za navijanje uparjal - planira	1	1 S		140	0	0
OV784S 0020		navijanje	1	1 S		140	0	0
OV784O 0030		Pripraviti celico, naviti cev	1	0 S		140	1	1
OV784A 0040		Spojiti celico in CPU	1	0 S		136	1	1
OV784A 0050		He detekcija	1	0 S		140	1	1

Slika 2.1: V tabelo izvožen opis tehnološkega postopka za izdelavo gospodinjskega aparata v podjetju Hisense Gorenje Europe.

2.1.2 Definicija problematike

V diplomski nalogi raziščemo možne metode izdelave delavníškega dnevnika oz. opisa tehnološkega postopka. Raziskati želimo prednosti in slabosti trenutnih metod sestavljanja teh dokumentov. V naslednji fazi dela želimo izdelati specializiran sistem za sestavljanje opisov tehnoloških postopkov. V ta sistem želimo vpeljati glasovnega pomočnika in raziskati, kako uporaba glasovnega pomočnika vpliva na uporabo sistema.

2.2 Funkcionalne zahteve

Razviti želimo mobilno aplikacijo, s katero bo lahko uporabnik upravljal z zbirkо delavníških dnevnikov. Za mobilno aplikacijo smo se odločili, saj je pametni telefon veliko prikladnejši na delovnem mestu kot npr. prenosni računalnik.

Uporabniku sistema bi radi omogočili, da lahko enostavno ustvarja delavniške dnevниke in jih dodaja v svojo zbirko. Želimo, da lahko uporabnik v delavniške dnevnike vstavlja korake, s katerimi opiše postopek dela. Ti koraki se morajo prikazovati v preglednem in urejenem seznamu. Za vsak delavniški dnevnik želimo imeti tudi povzetek dela in seznam možnih nevarnosti, ki pri delu pretijo delavcu.

Ker se lahko tehnološki postopki spreminjajo, želimo v delavniškem dnevniku omogočiti tudi naknadno brisanje in spreminjanje korakov.

Nekatere funkcionalnosti mobilne aplikacije želimo sprožiti prostoročno z glasovnim pomočnikom, saj bi tako sestavljač manjkrat prekinil delo. Med funkcionalnosti spadajo narekovanje koraka delavniškega dnevnika in odpiranje kamere za dodajanje fotografij.

Delavniške dnevnike želimo sinhronizirati na oddaljen strežnik, ki teče neodvisno od aplikacije, da se izognemo podvajanju datotek.

Uporabnik mora imeti tudi možnost izvoza svojih delavniških dnevnikov v standardne računalniške formate, ki jih lahko uporabi izven našega sistema.

2.3 Uporabljene tehnologije

2.3.1 .NET

.NET [3] je razvojna platforma, razvita s strani Microsofta. Obsega programske jezike, prevajalnike, orodja in knjižnice, ki omogočajo širok spekter primerov uporabnosti, hkrati pa se ohranja enovitost ozadne kode.

Tehnologije .NET ogrodja, ki smo jih uporabil v tej diplomski nalogi, so:

- .NET Core - odprtokodna platforma za razvoj spletnih storitev,
- Xamarin - ogrodje za razvoj mobilnih aplikacij za najpogosteje mobilne operacijske sisteme (Android, iOS).

.NET smo izbrali, ker je zelo dobro integriran z Amazonovim AWS API in je dobro dokumentiran.

2.3.2 Xamarin

Ogrodje Xamarin [15] je odprtakodno orodje za razvoj mobilnih aplikacij, ki ga razvija Microsoft.

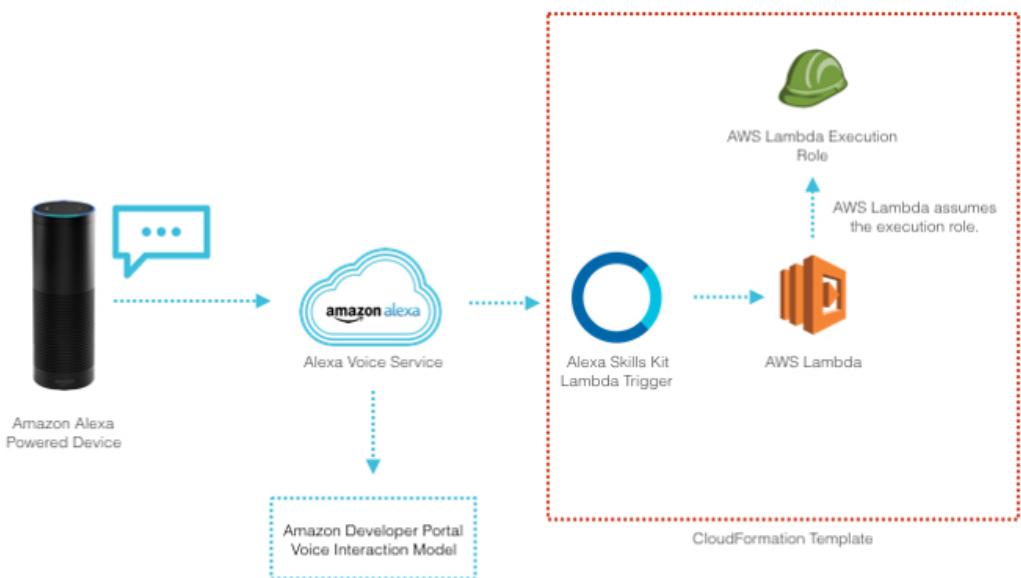
Z njim je mogoče deliti večino ozadne in ospredne kode med različnimi mobilnimi operacijskimi sistemi. Za ozadno kodo se uporablja .NET (C#), za ospredno kodo pa se uporablja XAML (Extensible Application Markup Language).

Xamarin smo izbrali, da smo lahko strežniški program in mobilno aplikacijo programirali v enakem programskem jeziku (C#).

2.3.3 Amazon Alexa

Amazon Alexa je glasovni pomočnik, razvit s strani podjetja Amazon [1]. Za Amazon Alexa smo se odločili, saj ponuja ogrodje za programiranje dodatnih funkcionalnosti, ki se imenuje Alexa Skills Kit. Alexa je enostavno integrirati z drugimi Amazonovimi spletnimi storitvami, ki smo jih uporabili v tej diplomski nalogi (AWS SQS, AWS Lambda).

Alexine osnovne funkcionalnosti lahko nadgradimo s programi, ki se jim reče „Skill-i“, v nadaljevanju „veščine“ (na sliki 2.2) [13]. Za razvoj in objavo Alexa veščine rabimo račun Amazon razvijalca (*ang. Amazon Developer Account*).



Slika 2.2: Arhitektura Alexine veščine (vir [11]).

Veščino sestavlja:

- **Poziv** - fraza, ki veščino zažene,
- **Namere** - fraze, ki jih veščina razpozna kot funkcije,
- **Krajišče** - omrežni vir, kjer se nahaja ozadna koda veščine.

Alexa zajame glasovni ukaz in glasovni posnetek pošlje na Amazonov Alexa Voice Service. Ta storitev s pomočjo glasovnega razpoznavnega modela prepozna uporabnikove ukaze in pošlje poseben zahtevek na krajišče. Krajišče je lahko druga Amazonova storitev (npr. AWS Lambda), storitev na Microsoftovem Azure strežniku, ali naš lasten strežnik, dostopen preko javne domene in zaščiten z overjenim SSL certifikatom.

Ko krajišče obdela zahtevo, odgovor pošlje nazaj na Alexa Voice Service. Ta se iz Alexa Voice Service pošlje nazaj na Alexa, ki uporabniku „izgovori“ prejeto sporočilo.

2.3.4 Amazon Web Services

Amazon Web Services (v nadaljevanju AWS) je skupek oblăčnih storitev, ki jih ponuja podjetje Amazon. Ponuja integracijo s pogostimi programskimi jeziki in ogrodji kot so Java, .NET, Python in Node.js preko storitve AWS API. Pri tej diplomski nalogi smo se osredotočili na storitvi AWS Lambda in AWS Simple Queue Service.

Simple Queue Service

AWS Simple Queue Service (v nadaljevanju SQS) je sistem za pošiljanje besedilnih sporočil med odjemalci preko Amazonovih strežnikov [5]. To storitev smo uporabili za komunikacijo med Amazon Alexa in strežnikom implemen-tiranega sistema.

Za hranjenje sporočil je potrebno registrirati SQS vrsto. Uporabili smo SQS vrsto tipa FIFO (first in, first out) [9], pri kateri je zagotovljeno, da na strežnik prejmemo vsa sporočila v istem vrsttem redu, kot so bila na SQS vrsto poslana.

Lambda

Ozadno kodo Alexa veščine smo gostili na platformi AWS Lambda, ki je storitev za gostovanje dogodkovno vodene ozadne kode [4]. Za to platformo smo se odločili zaradi dobre integracije z Alexa Skill Kit in razvojnim orodjem Visual Studio.

2.4 Obstojče rešitve

2.4.1 Papir in pisalo

Najstarejša metoda za izdelavo delavniškega dnevnika je zapis na list papirja (slika 2.3).

Opis tehnološkega postopka

Avtor: ANEJ LEKŠE

Datum: 18.8.2020

Naslov:	ČIŠČENJE TIPKOVNICE
Opis:	ČIŠČENJE MEHANSKE TIPKOVNICE
Nevarnosti:	/

Korak	Opombe	Trajanje
PRIPRAVI DELOVNO MIZO		5 MIN
PRIPRAVI TIPKOVNICO		1 MIN
ODSTRANI POKROVE TIPK.	ORODJE JE PRILOŽENO V ŠKATLI	10 MIN
POKROVE TIPK NAMOČI V MILNICO	PUSTI VSAJ 10 MIN	minimalno 10 MIN
OBRISI POKROVE TIPK DO POVSEM SUHEGA		30 MIN
DAJ POKROVE TIPK NAZAJ NATIPK.		8 MIN

Slika 2.3: Opis tehnološkega postopka za čiščenje tipkovnice napisan ročno na obrazec.

Prednosti uporabe papirja in pisala pred našim sistemom so:

- pri delu izdelovalec ne potrebuje računalnika in
- cenovna ugodnost.

V primerjavi z našim sistemom so slabosti te metode:

- prostorske omejitve,
- problematično dopisovanje in urejanje obstoječih korakov,
- zahtevnejše arhiviranje od računalniških datotek,
- občutljivost papirja na fizične poškodbe (trganje, mečkanje, vnetljivost...),
- čitljivost pisave.

2.4.2 Pisarniški programi

Opis tehnološkega postopka lahko izdelamo v pisarniških programih, kot so na primer Microsoft Word in LibreOffice Writer.

Ta pristop reši nekatere slabosti uporabe papirja in pisala za sestavljanje opisa tehnološkega postopka. Prednosti tega pristopa so:

- enostavno dodajanje in urejanje korakov,
- možnost dodajanja slikovnega gradiva,
- pisarniški programi omogočajo izpis dokumenta na tiskalnik, če želimo imeti dokument natisnjen na papirju.

Kljub temu uporaba te metode v primerjavi z našim sistemom prinese nove slabosti:

- če imamo dokument shranjen na več mestih, moramo ob spremembah zagotoviti, da se posodobijo vsi shranjeni dokumenti,
- slikovno gradivo je vezano na dokument; če na novo zajamemo sliko, ki smo jo vstavili v dokument, jo moramo spremeniti tudi znotraj dokumenta.

V sklopu diplomske naloge smo napisali preprost opis tehnološkega postopka (slika 2.4) s programom LibreOffice Writer [7]. Dokument brez vsebine se lahko pri naslednjih opisih uporabi kot obrazec za sestavljanje delavnih dnevnikov.

Opis tehnološkega postopka

Avtor: Anej Lekše

Datum: 12. 8. 2020

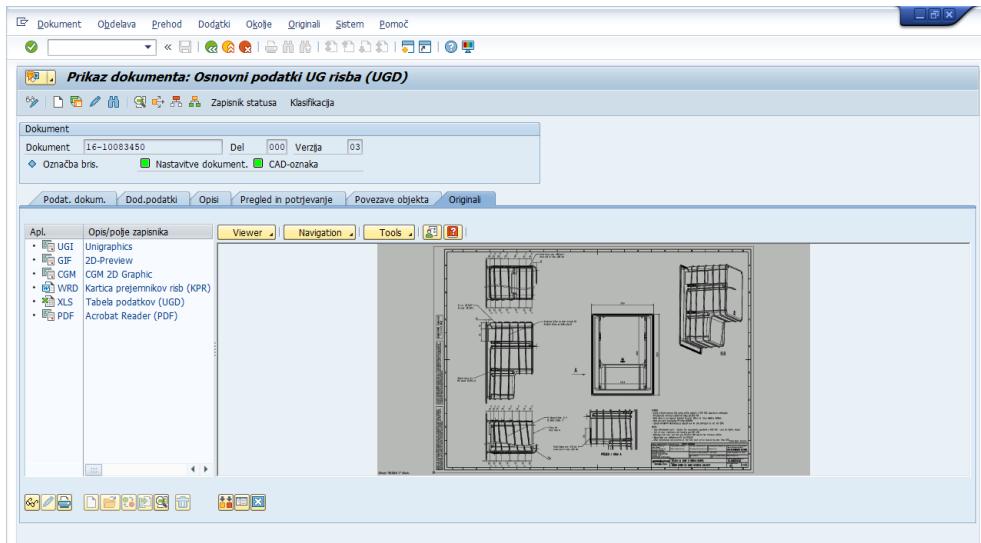
Naslov:	Čiščenje tipkovnice
Opis:	Čiščenje mehanske tipkovnice
Nevarnosti:	Ni nevarnosti

Korak	Opombe	Trajanje
Pripravi delovno mizo in tipkovnico		5 min
Odstrani pokrove tipk	Uporabi orodje, ki je bilo priloženo ob tipkovnici	10 min
Pokrove tipk namoči v milnico	Pusti v vodi vsaj 10 minut	10 min
Obrisi pokrove tipk do suhega		30 min
Vstavi pokrove tipk na svoja mesta		8 min

Slika 2.4: Preprost opis tehnološkega postopka za čiščenje tipkovnice napisan v pisarniškem programu.

2.4.3 Specializirani moduli za poslovne informacijske sisteme

Podjetja, kot so Hisense Gorenje Europe, za sestavljanje delavnih dnevnikov uporabljajo specializirane module za lastne informacijske sisteme. Kot primer je prikazan korak opisa tehnološkega postopka v informacijskem sistemu SAP (na sliki 2.5).



Slika 2.5: Prikazan korak opisa tehnološkega postopka s programom SAP.

Do opisov tehnološkega postopka se lahko dostopa iz računalnikov na delovnih mestih. Opis tehnološkega postopka v sistemu SAP sestavlja:

- podatki o izdelku,
- opisi korakov,
- definicija kontrolnih postopkov in pregleda,
- CAD izris izdelka,
- dodatne opombe.

Prednosti sistema SAP pred našim sistemom so:

- sistem SAP je tesno povezan s proizvodno linijo in prilagodljiv za potrebe proizvodnega obrata, ki ga uporablja,
- podatki se hranijo na osrednjem strežniku, varnostna kopija pa se inkrementalno dela na geografsko ločen rezervni strežnik.

Takšni sistemi, v primerjavi z našim, predstavljajo naslednje slabosti:

- zahtevajo dovršeno proizvodno infrastrukturo, ki jo sistem rabi za optimalen izkoristek in
- visoka cena, ki je potrebna za implementacijo takega sistema v proizvodno linijo.

Poglavlje 3

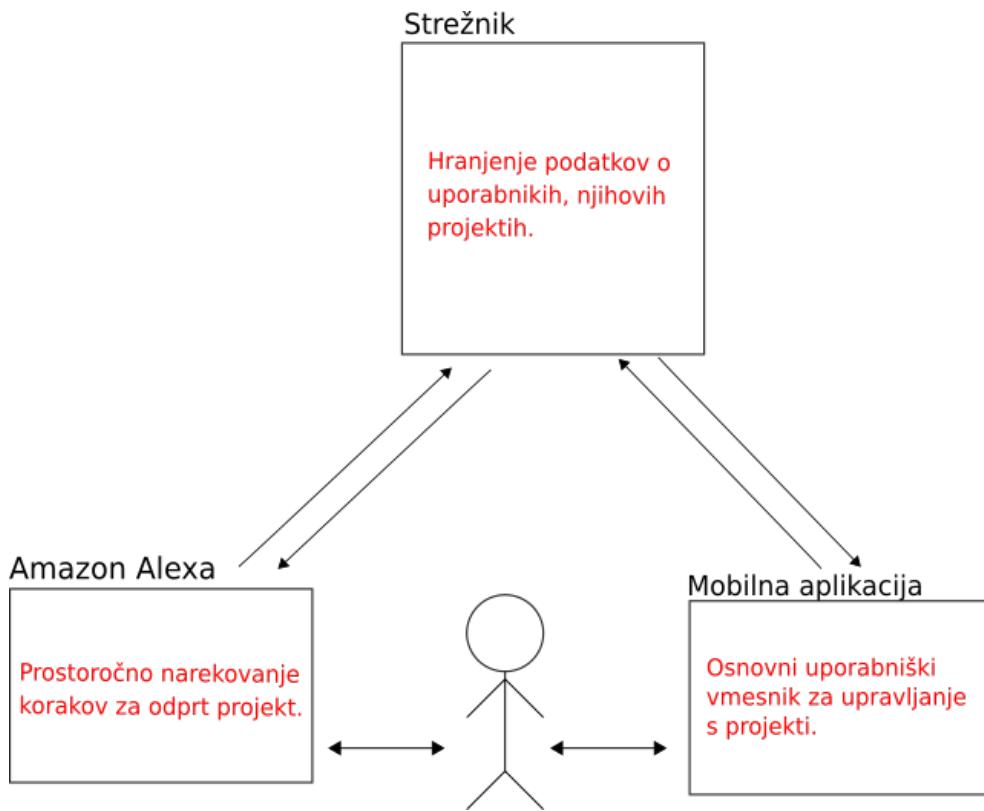
Načrtovanje in razvoj sistema

3.1 Definicija funkcionalnosti

Sistem, ki smo ga v sklopu diplomske naloge implementirali (na sliki 3.1), uporabniku omogoča vodenje delavniških dnevnikov (v nadaljevanju imenovanih tudi *projektov*). Služil bo kot alternativa pisanju delavniških dnevnikov s pisarniškimi programi. Vsak projekt ima zbirko *korakov*. Korak opisuje posmezno dejavnost tehnološkega postopka, ki ga želimo zabeležiti z delavniškim dnevnikom.

Implementiran sistem sestavlja:

- mobilna aplikacija,
- glasovni pomočnik in
- strežnik.



Slika 3.1: Arhitektura sistema.

Osnovni uporabniški vmesnik sistema je mobilna aplikacija. Z aplikacijo se uporabnik prijavi v sistem s svojim uporabniškim računom, ustvarja, odpira in ureja svoje delavniške dnevnike.

Delavniški dnevnik oz. projekt sestavljajo naslov, opis dela, opis možnih nevarnosti in seznam korakov postopka. Korake sestavljajo naslov, opis in predvideno trajanje koraka.

Seznam korakov postopka mora podpirati dodajanje novih korakov, posodabljanje vsebine obstoječih korakov, brisanje obstoječih korakov in spremnjanje vrstnega reda korakov. Uporabnik lahko tudi posname fotografije in jih doda v odprt projekt kot korak. Vsi podatki, ki jih uporabnik vnese preko mobilne aplikacije in glasovnega pomočnika, se hranijo na strežniku.

Glasovnega pomočnika smo uporabili kot razširitev uporabniškega vme-

snika. Z implementacijo glasovnega pomočnika smo želeli doseči:

- narekovnaje besedilnega koraka projekta,
- prostoročno odpiranje obrazca za dodajanje koraka v projekt in
- prostoročno odpiranje kamere na mobilni napravi.

3.1.1 Izboljšave z glasovnim pomočnikom

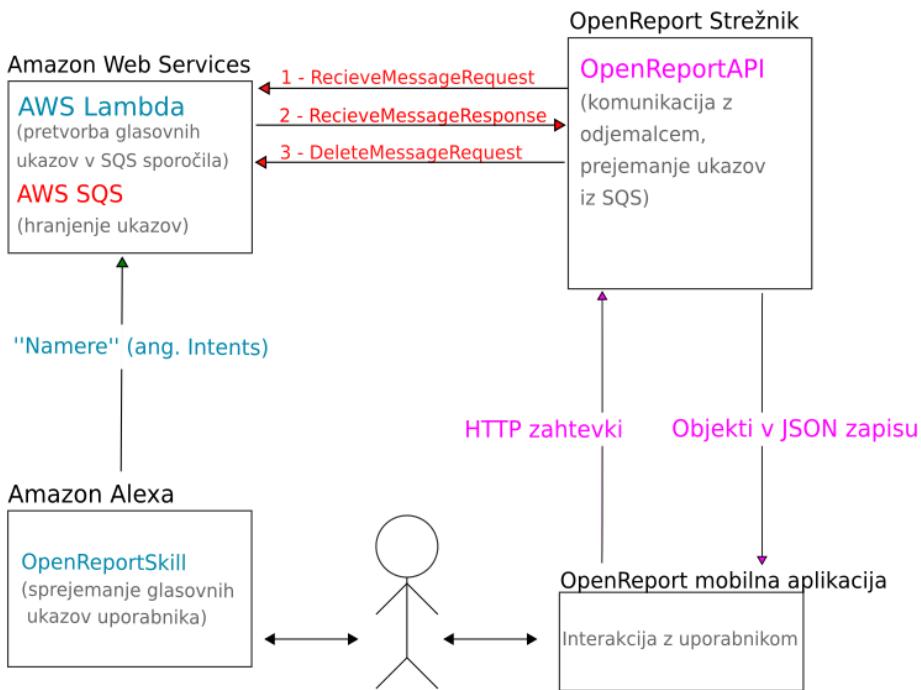
Komunikacija z glasovnim pomočnikom je smiseln način interakcije z računalniškim sistemom v situaciji, kjer uporabnik nima prostih rok ali je fizično oddaljen od naprave.

Raziskava iz leta 2018 [2] je pokazala izboljšano učinkovitost pri delu raziskovalcev v kemijskem laboratoriju, v katerega so integrirali glasovne pomočnike. Namen raziskave je bil preizkus praktične uporabnosti glasovnih pomočnikov za branje laboratorijskih postopkov in glasovno upravljanje laboratorijskih instrumentov. Pozitivni rezultati bi lahko bili ključnega pomena za slabovidne člane laboratorijev. Kot glasovnega pomočnika so uporabili Amazon Alexa. Prepoznavanje govora in ukazov je bilo konsistentno in hitro ne glede na spol uporabnika. Motnje pri razpoznavanju je povzročal večinoma hrup v ozadju. Povprečna uspešnost prepoznavane ukazov je bila 95%.

3.2 Načrt sistema za sestavljanje opisov teh-noloških postopkov

Implementirali smo arhitekturo, ki je predstavljena v poglavju 3.1. Sistem smo poimenovali OpenReport (na sliki 3.2), ki ga sestavlja:

- mobilna aplikacija,
- strežniški program in
- Amazon Alexa.



Slika 3.2: Visokonivojski načrt sistema.

Strežnik v podatkovni bazi hrani uporabniške račune in opise tehnoloških postopkov. Uporabnik lahko do podatkov dostopa preko mobilne aplikacije, ki s strežnikom komunicira preko API. Glasovnega pomočnika smo uporabili kot dodatek k mobilni aplikaciji. Omogoča glasovno upravljanje aplikacije in narekovanje korakov.

Preko mobilne aplikacije uporabnik lahko:

- opravi registracijo in prijavo,
- ustvari nov delavniški dnevnik,
- odpre obstoječe delavniške dnevниke,
- ustvarja in ureja korake delavniškega dnevnika,
- zajema slike in jih vstavlja v delavniški dnevnik,

- briše korake delavniškega dnevnika,
- ureja vrstni red korakov delavniškega dnevnika.

Za glasovnega pomočnika Alexa smo razvili veščino, s katero uporabnik lahko:

- v odprto poročilo vstavi dobesedno narekovani korak,
- odpre obrazec za dodajanje novega besedilnega koraka,
- odpre kamero in obrazec za dodajanje koraka s fotografijo.

3.3 Strežnik

Osrednja komponenta sistema OpenReport je strežnik. Zadolžen je za delo s podatki uporabnikov in delavniškimi dnevniki. Odjemalci, ki so nanj povezani, služijo zgolj kot „uporabniški vmesnik“ za zajem in prikaz podatkov na strežniku.

Osnovne naloge strežnika so:

- avtentikacija uporabnikov,
- avtorizacija uporabnikovih zahtev,
- komunikacija z glasovnim pomočnikom,
- hranjenje podatkov o uporabnikih in delavniških dnevnikih,
- ponujanje vmesnika, ki ga lahko odjemalci uporabijo za operacije nad delavniškimi dnevniki.

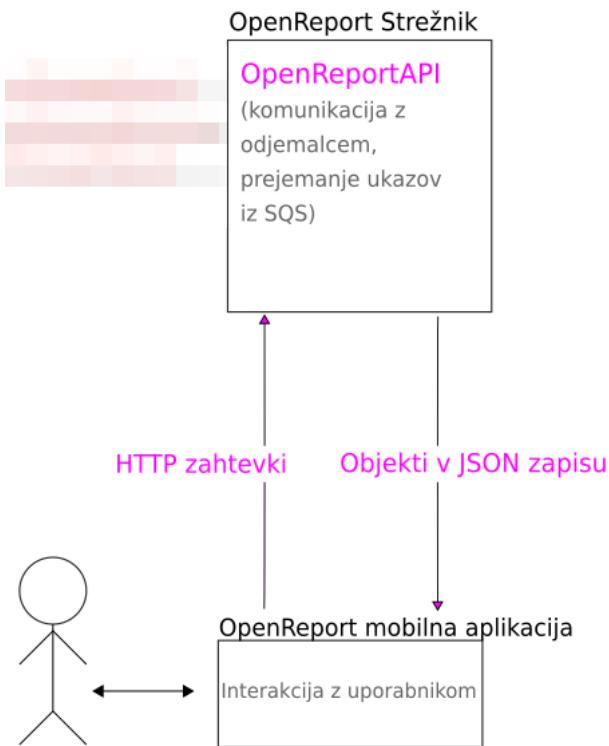
Podatke hranimo na strežniku, da lahko do podatkov dostopamo iz različnih naprav preko enotnega vmesnika. To nam omogoča enostavno dodajanje odjemalcev in odpravi podvajanje podatkov med napravami.

S strežnikom komunicirata mobilna aplikacija in glasovni pomočnik. Podatke hranimo v podatkovni bazi, ki jo hrani SQL Server.

3.3.1 Komunikacija z odjemalci

Strežnik z odjemalci komunicira preko API zaradi enostavnosti implementacije in možnosti širjenja nabora odjemalcev v prihodnosti.

V našem primeru je bil osnovni odjemalec mobilna aplikacija. Odjemalec na definirane funkcijске URL-je (slika 3.5) strežnika pošlje HTTP zahtevke. Če so zahtevki pravilno oblikovani, strežnik izvede predvideno funkcijo in rezultat te funkcije pošlje kot HTTP odgovor nazaj odjemalcu. Komunikacija med strežnikom in odjemalcem je prikazana z roza barvo na sliki 3.3.



Slika 3.3: Komunikacija strežnika z odjemalcem.

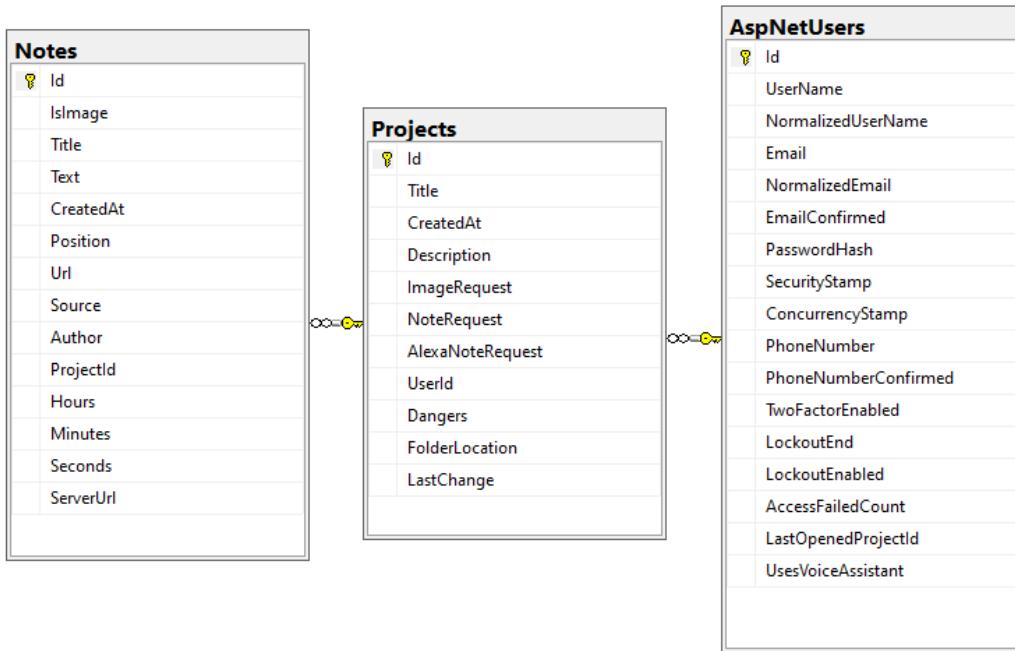
3.3.2 Podatkovni model

Podatke smo razdelili v tri entitete:

- uporabnik,
- delavniški dnevnik,
- korak delavniškega dnevnika.

Uporabnik lahko sestavlja svoje delavniške dnevnike, vsak delavniški dnevnik pa je sestavljen iz zbirke korakov.

Podatkovna baza hrani tabele „AspNetUsers“, „Projects“ in „Notes“ (slika 3.4). Vsak uporabnik lahko ima 0 ali mnogo delavniških dnevnikov. Vsak projekt ima lahko 0 ali mnogo korakov.



Slika 3.4: ER model podatkovne baze.

Uporabnik ima uporabniško ime, katerega uporabi za prijavo, in geslo s katerim dokaže, da je to res on.

Vsek delavniški dnevnik ima naslov, opis in enolični identifikator uporabnika, ki si dnevnik lasti.

Vsek korak delavniškega dnevnika ima naslov, opis, podatke o trajanju in enolični identifikator projekta, ki si korak lasti.

3.4 Implementacija strežnika in mobilne aplikacije

Komunikacija med odjemalcem in strežnikom poteka preko spletnega API (slika 3.5), ki ga ponuja strežnik.

Ker lahko sistem uporablja več uporabnikov, smo implementirali sistema za avtentikacijo uporabnikov in avtorizacijo zahtev. Avtentikacija z upo-

rabniškim imenom in geslom omogoča preverjanje identitete. Ko se vzpostavi zaupanje, se uporabniku dodeli avtorizacijski žeton.

Project		Users	
GET	/api/v1/projects	PUT	/api/v1/users/requestva
POST	/api/v1/projects	PUT	/api/v1/users/releaseva
GET	/api/v1/projects/{id}	POST	/api/v1/users/current
DELETE	/api/v1/projects/{id}		
GET	/api/v1/projects/{id}/notes		
GET	/api/v1/projects/{id}/nonotes		
POST	/api/v1/projects/{id}/addnote		
POST	/api/v1/projects/{id}/addimage		
POST	/api/v1/projects/{id}/export/text		
POST	/api/v1/projects/{id}/export/html		
PUT	/api/v1/projects/last/closerequests		
PUT	/api/v1/projects/{pid}/update/{nid}		
PUT	/api/v1/projects/{pid}/{nid}/{positions}		
DELETE	/api/v1/projects/{pid}/delete/{nid}		

Slika 3.5: Seznam razvitih in dokumentiranih API funkcij.

3.4.1 Pristop razvoja mobilne aplikacije

Pristop razvoja aplikacije, ki smo ga uporabili za programiranje aplikacije, se imenuje „Model View View-Model“ (v nadaljevanju MVVM). Pri tem pristopu aplikacijo razdelimo na tri dele (slika 3.6).

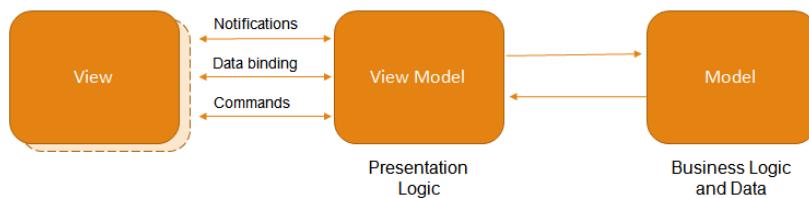
V **Modelu** definiramo elemente naše podatkovne logike (opis tehnološkega postopka, korak, uporabnik).

View je uporabniški vmesnik, ki ga vidi uporabnik.

View-Model se uporablja, da se poveže funkcije uporabniškega vmesnika in podatkovne logike ter po potrebi preoblikuje podatke.

Rezultat upoštevanja tega pristopa je čista koda, ki nima prepletene elementov med ozadno kodo in uporabniškim vmesnikom. „Model“ vsebuje le abstrakcijo naših podatkov in podatkovno logiko. Ti podatki se v „View-Model-u“ pretvorijo v obliko, ki bo prikazana uporabniku. „View“ nato prikaže pripravljene podatke uporabniku v obliki grafičnih elementov.

Na spodnji sliki lahko vidimo oris pristopa MVVM (slika 3.6).



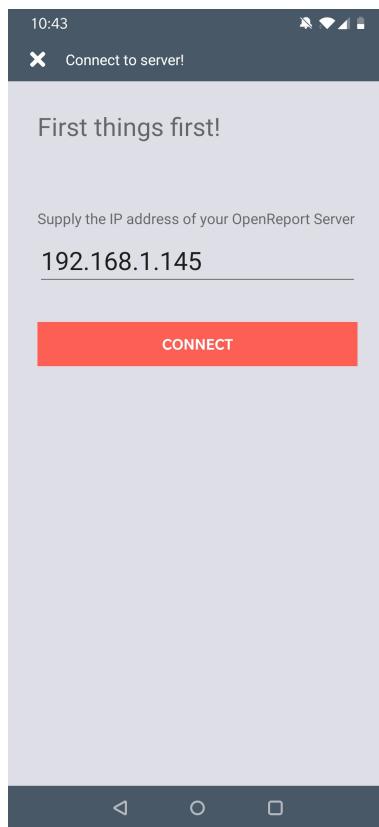
Slika 3.6: Shema pristopa MVVM (vir [10]).

3.4.2 Povezava na strežnik

Ko uporabnik zažene mobilno aplikacijo OpenReport, se mora najprej povezati na strežnik. Na strani za povezavo (slika 3.7) mora vnesti IP naslov strežnika. Vnešen IP bomo v tem podpoglavlju označevali z `{naslov}`. Ob pritisku na gumb Connect ozadna koda preveri, ali na tem IP naslovu teče OpenReport strežnik tako, da na „`{naslov}/openreportapi/api/connect`“ pošlje HTTP zahtevek GET.

Če na podanem naslovu teče OpenReport strežnik, ta odjemalcu vrne odgovor `true`. Ozadna koda aplikacije javi napako, če na poslano zahtevo ne dobi odgovora, ali če naprava s tem IP ne obstaja.

Ob uspešni povezavi na strežnik ozadna koda aplikacije uporabi naslov „`{naslov}/openreportapi/api`“ kot predpono vseh nadaljnjih zahtevkov na strežnik.

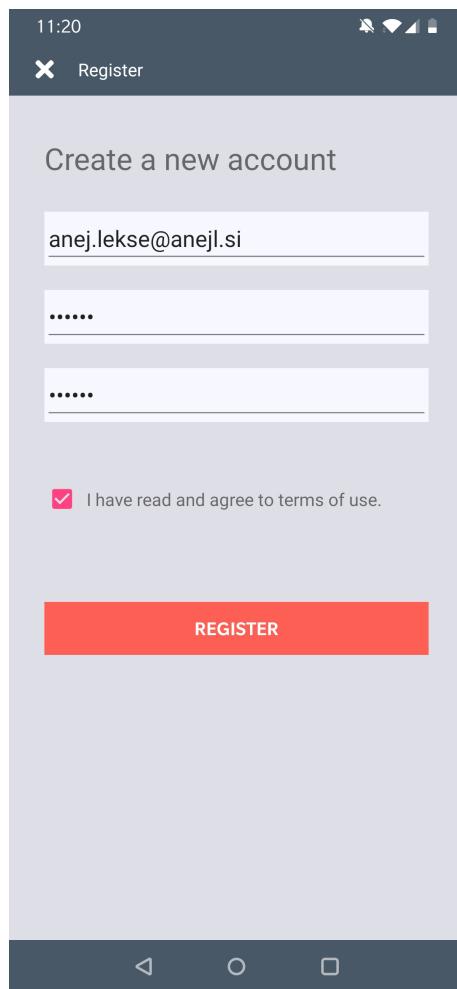


Slika 3.7: Povezovanje mobilne aplikacije s strežnikom.

3.4.3 Registracija in prijava

Vse operacije nad uporabnikovimi delavniki morajo biti avtorizirane, zato mora uporabnik vzpostaviti zaupanje s strežnikom. To uporabnik storí tako, da se v sistem prijavi s svojim uporabniškim imenom in geslom. Ob prvi uporabi se mora uporabnik registrirati (slika 3.8).

Ob uspešni avtentikaciji uporabnik od strežnika prejme avtorizacijski žeton, ki mu dovoljuje dostop do njegovih delavnih dnevnikov. Aplikacija doda avtorizacijski žeton vsem nadaljnji HTTP zahtevkom na strežnik.



Slika 3.8: Registracijska stran v mobilni aplikaciji.

Pri registraciji mobilna aplikacija na strežnik pošlje objekt razreda `RegisterUserRequest` na funkcijski URL „/identity/register“ preko metode POST. V objektu `RegisterUserRequest` se nahajata njegov e-mail naslov in geslo v čisti obliki.

```
RegisterUserRequest {  
    string Email;  
    string Password;  
}
```

Na strežniku zahtevek obdela avtentikacijska storitev. Storitev preveri, ali je e-naslov v polju `Email` že bil registriran. Če je bil, se zabeleži napaka in nadaljnja registracija se prekine. Če ta uporabnik ne obstaja, se ustvari nov objekt razreda `User`. Polje `Password` se šifrira in se skupaj s poljem `Email` zapiše v ta objekt. Objekt se zapiše v podatkovno bazo v tabelo `Users`. Uporabnik pri registraciji dobi tudi svoj enoličen identifikator `Id`.

Objekt `AuthFailedResponse` se vrne odjemalcu ob napaki med avtentikacijo. Vsebuje seznam vseh zabeleženih napak.

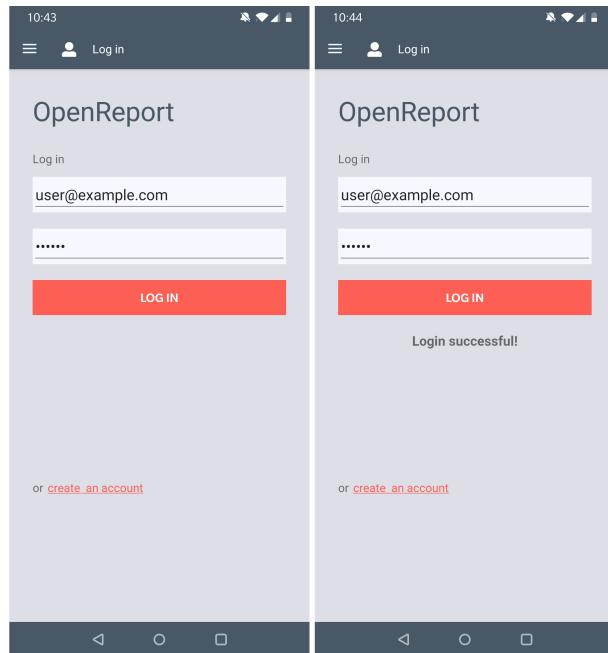
```
AuthFailedResponse {
    IEnumerable<string> Errors;
}
```

Objekt razreda `AuthSuccessResponse` se pošlje odjemalcu ob uspešni registraciji.

```
AuthSuccessResponse {
    string UserId;
    string Token;
}
```

V polje `Token` razreda `AuthSuccessResponse` se zapiše avtorizacijski žeton. Žeton je tipa JWT ali JSON Web Token [16]. Sestavlja ga e-mail uporabnika, uporabnikov enolični identifikator, čas zapada žetona in tip simetričnega šifriranja uporabljenega za šifriranje žetona.

Žeton v objektu razreda `AuthSuccessResponse` se pošlje odjemalcu.



Slika 3.9: Prijavna stran pred poskusom prijave (levo), in po uspešnem poskusu prijave (desno).

Prijava (na sliki 3.9) poteka podobno. Odjemalec na strežnik pošlje objekt razreda `LoginUserRequest` na funkcionalni URL „/identity/login“. Objekt `LoginUserRequest` vsebuje e-mail naslov in geslo v čisti obliki.

```
(LoginUserRequest {
    string Email;
    string Password;
})
```

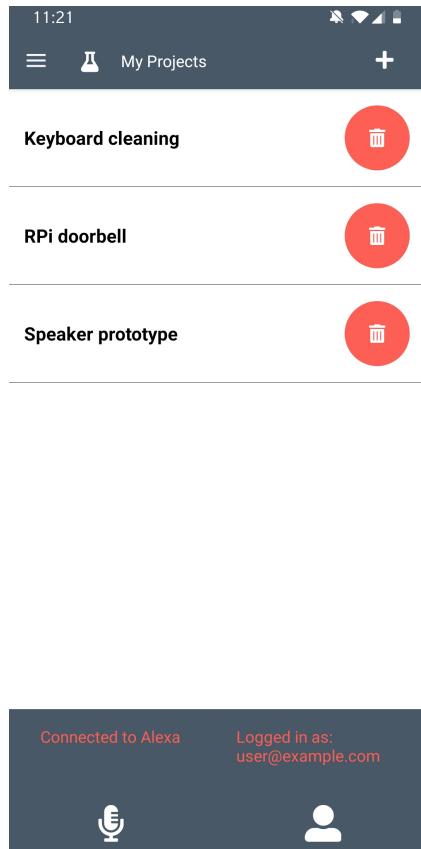
Avtentifikacijska storitev preveri, ali uporabnik s tem e-naslovom že obstaja. Če uporabnik ne obstaja ali pa je šifrirano geslo v podatkovni bazi drugačno kot to, kar je v polju `Password`, zabeležimo napake in prijava se prekine. Strežnik odjemalcu pošlje objekt razreda `AuthFailedResponse` s seznamom napak.

Če uporabnik s podanim e-naslovom obstaja in se šifrirano geslo iz polja `Password` ujema z gesлом v podatkovni bazi, strežnik odjemalcu pošlje avtorizacijski žeton v objektu razreda `AuthSuccessResponse`.

3.4.4 Operacije z delavniškimi dnevnikmi

Po prijavi lahko uporabnik odpre stran s seznamom njegovih delavniških dnevnikov, imenovano Dashboard (na sliki 3.10). Na tej strani se prikažejo:

- seznam vseh njegovih projektov,
- gumbi za izbris posameznega projekta,
- gumb za dodajanje novega projekta,
- gumb za zahtevanje ali sprostitev glasovnega pomočnika.

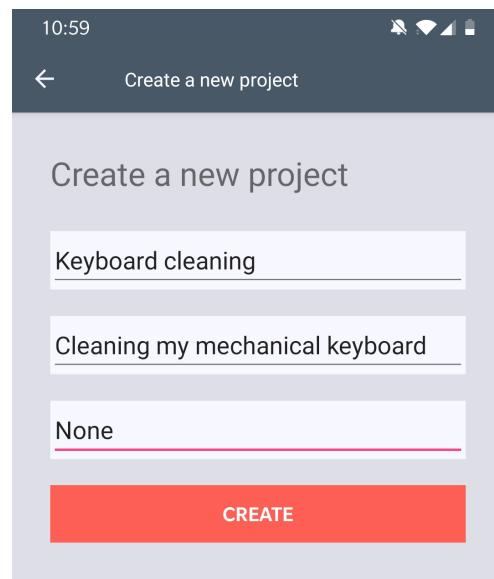


Slika 3.10: Stran Dashboard v mobilni aplikaciji.

Uporabnik nov delavniški dnevnik ustvari tako, da pritisne na gumb z ikono + na strani Dashboard. Odpre se obrazec za ustvarjanje projektov (na sliki 3.11), kamor vnese naslov, kratek opis projekta in opis možnih nevarnosti. Te podatke ozadna koda aplikacije zapiše v objekt razreda `CreateProjectRequest`.

```
CreateProjectRequest {  
    string Title;  
    string Description;  
    string Dangers;  
}
```

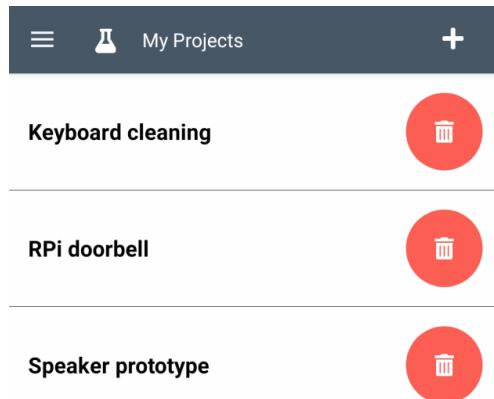
`CreateProjectRequest` aplikacija pošlje na URL „/projects/create“ preko metode POST. Strežnik nato ustvari nov dnevnik s podatki iz prejete zahteve. Odjemalcu se kot odgovor pošlje ustvarjen objekt tega projekta.



Slika 3.11: Obrazec za ustvarjanje novega projekta v aplikaciji.

Razred Project izgleda tako:

```
Project {  
    int Id; // unikatni identifikator  
    string Title;  
    string Description;  
    string Dangers;  
    IEnumerable<Note> Notes; // seznam korakov  
    ...  
}
```



Slika 3.12: Seznam delavniških dnevnikov v mobilni aplikaciji.

Avtentificiran uporabnik lahko do svojih delavniških dnevnikov dostopa tako, da s seznama dnevnikov (na sliki 3.12) izbere želenega. Aplikacija pošlje avtoriziran GET zahtevek na URL „/projects/{id}“. Polje {id} mora biti enolični identifikator projekta. Strežnik uporabniku vrne kopijo objekta delavniškega dnevnika in pripadajočih korakov.

Uporabnik lahko delavniški dnevnik izbriše tako, da pritisne na gumb z ikono smetnjaka (na sliki 3.12) ob naslovu odvečnega dnevnika. Aplikacija pošlje DELETE zahtevo na URL „/projects/{id}“. Če zahteva ni avtorizirana z ustreznim žetonom, se projekt ne izbriše.

3.4.5 Operacije s koraki delavniškega dnevnika

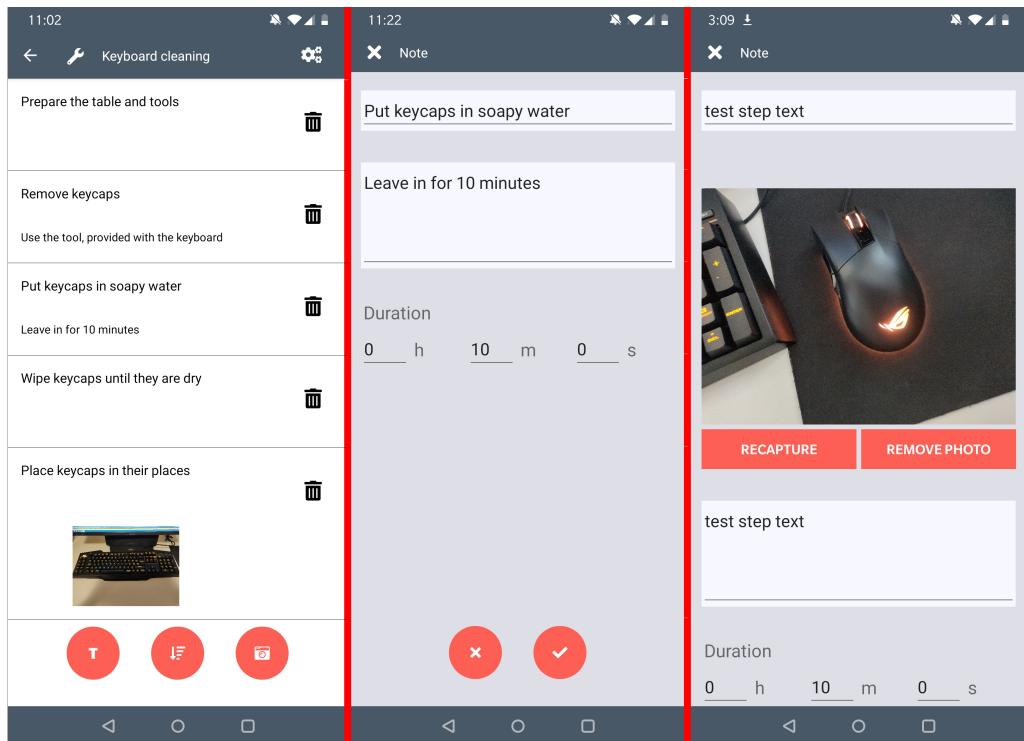
Korak delavniškega dnevnika opisuje eno od nalog v tehnološkem postopku.

Delavniški dnevnik lahko vsebuje tudi slikovno gradivo. Korake smo ločili na slikovne in besedilne. Besedilni korak vsebuje naslov, besedilo in trajanje naloge, ki jo opisuje; slikovni pa vsebuje tudi fotografijo.

Sistem podpira dodajanje in brisanje korakov, spreminjanje vsebine posameznega koraka in spremenjanje vrstnega reda korakov.

Dodajanje besedilnega koraka

Uporabnik v odprt projekt besedilni korak vstavi s pritiskom na gumb s črko T (na sliki 3.13 levo). Odpre se obrazec za dodajanje koraka (na sliki 3.13 na sredini). Obrazec ima polja za naslov, besedilo in trajanje.



Slika 3.13: Stran za pregled delavnškega dnevnika in obrazci za dodajanje besedilnih in slikovnih korakov v mobilni aplikaciji.

Vsebino teh polj ozadna koda aplikacije zapiše v objekt razreda Note. Naslov koraka se zapiše v polje Title, opis v polje Text, trajanje pa se zapiše v polja Hours, Minutes in Seconds.

```
Note {
    ...
    string Title;
    string Text;
```

```

    int Hours;
    int Minutes;
    int Seconds;
    ...
}
```

Ustvarjen objekt razreda `Note` aplikacija pošlje na URL „/projects/{id}-
/addnote“ preko metode POST. Polje `{id}` mora biti enolični identifikator
projekta, ki mu želimo dodati korak.

Strežnik prejet korak vstavi v projekt. Aplikaciji v odgovor vrne kopijo
ustvarjenega koraka na strežniku in ga doda svoji zbirki korakov trenutno
odprtrega projekta.

Dodajanje slikovnega koraka

Za hranjenje besedilnih in slikovnih korakov smo zaradi preprostosti imple-
mentacije uporabili isti razred (`Note`). Besedilni in slikovni korak se ločita
v vrednosti boolean zastavice `IsImage`. Besedilni korak ima to polje nasta-
vljeno na vrednost `false`, slikovni pa na `true`.

```

Note {
    bool IsImage;
    string Title;
    string Text;
    int Hours;
    int Minutes;
    int Seconds;
    string Url; // Lokacija slike na odjemalcu
    string ServerUrl; // Lokacija slike na strežniku
    ...
}
```

Uporabnik slikovni korak vstavi v projekt s pritiskom na gumb z ikono
kamere (na sliki 3.13 levo).

Odpri se privzeta aplikacija za kamero, s katero uporabnik zajame fotografijo. Po zajemu se odpre obrazec za dodajanje koraka (na sliki 3.13 desno). Fotografija se zašifrira v znakovni niz po metodi Base64. Vsebina vnosnih polj obrazca za naslov, besedilo in trajanje koraka se zapiše v objekt razreda Note.

Korak in šifrirana fotografija se zapišeta v zahtevek UploadImageRequest.

```
UploadImageRequest {
    Note Note;
    string ImageString;
}
```

Polje ImageString vsebuje v znakovni niz šifrirano fotografijo. Polje Note vsebuje korak delaviškega dnevnika. Ta objekt aplikacija poslje preko POST metode na URL „/projects/{id}/addimage“ (slika 3.14).

```
public async Task<Note> AddImageNoteToProject(Project p, Note n, string encodedimage)
{
    SimpleAPIRequest<Note> r = new SimpleAPIRequest<Note>($"{_serverUrl}/projects/{p.Id}/addimage", _client);

    var payload = new UploadImageRequest
    {
        Note = n,
        ImageString = encodedimage
    };

    return await r.Send(RequestType.Post, payload);
}
```

Slika 3.14: Koda za ustvarjanje in pošiljanje zahtevka za dodajanje slikovnega koraka v aplikaciji.

Strežnik prejet korak vstavi v projekt in dešifrira sliko iz znakovnega niza nazaj v slikovno datoteko. Slikovno datoteko nato hrani v svojem datotečnem sistemu. Lokacija fotografije v datotečnem sistemu strežnika se zapiše v polje ServerUrl.

Aplikaciji v odgovor vrne kopijo ustvarjenega koraka na strežniku, ki ga aplikacija doda zbirkni korakov trenutno odprtrega projekta.

Urejanje korakov v projektu

Uporabnik lahko korake v delavnkih dnevnikih posodablja. Spreminja lahko njihovo vsebino, naslov in trajanje. Obrazec za urejanje koraka odpre tako, da v seznamu korakov projekta (na sliki 3.13 levo) pritisne na besedilo koraka, ki ga želi posodobiti.

Uporabnik korak posodobi s PUT zahtevo na URL „/projects/{pid}/-update/{nid}“. Polje {pid} je enolični identifikator projekta, ki si lasti korak, {nid} pa enolični identifikator koraka, ki ga želimo posodobiti. Telo zahteve mora vsebovati objekt razreda `Note`, ki ga je uporabnik posodobil.

Uporabnik slikovni korak posodobi s PUT zahtevo na naslov „/projects-{pid}/update/{nid}/image“. Polje {pid} je enolični identifikator projekta, ki si lasti korak, {nid} pa enolični identifikator koraka, ki ga želimo posodobiti. Telo te zahteve mora vsebovati zahtevek `UploadImageRequest`. V `UploadImageRequest` mora biti polje `Note` objekt, ki ga želimo posodobiti. Polje `ImageString` mora biti šifrirana slika, ki bo zamenjala prejšnjo sliko. Slika mora biti šifrirana po metodi Base64.

Brisanje korakov v projektu

Korak lahko uporabnik iz projekta izbriše tako, da pritisne gumb z ikono črnega smetnjaka ob koraku (na sliki 3.13 levo).

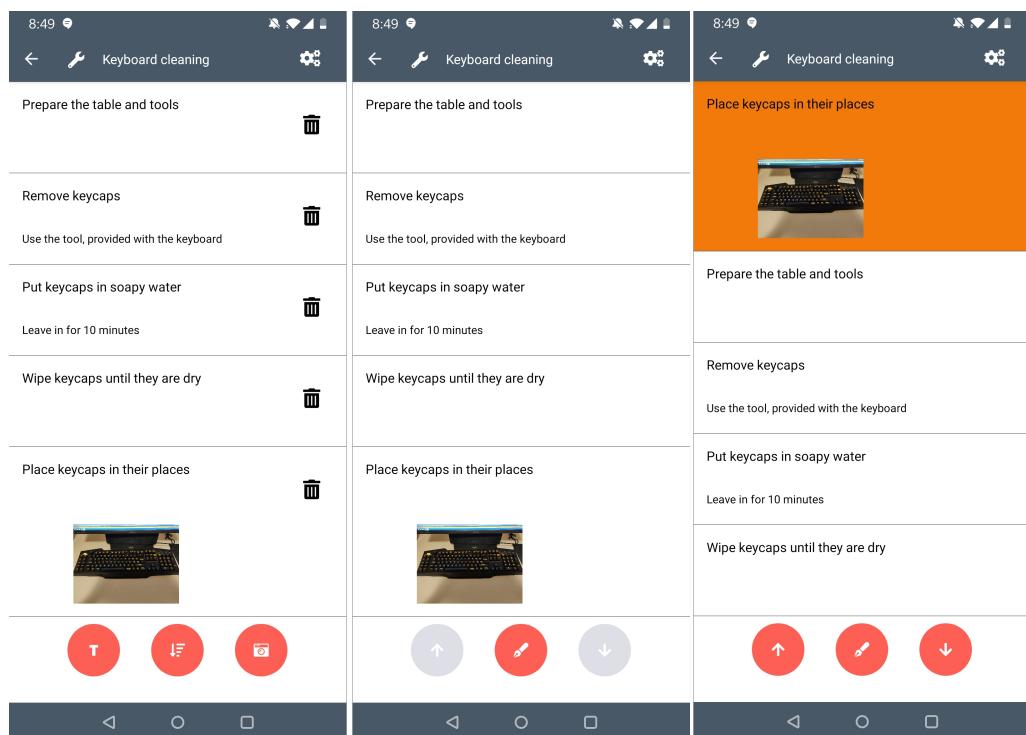
Ozadna koda aplikacije pošlje DELETE zahtevo na URL „/projects/-{pid}/delete/{nid}“. Polje je {pid} enolični identifikator projekta, v katerem se nahaja korak, {nid} pa enolični identifikator koraka, ki ga želimo izbrisati.

Če ima korak nastavljeno polje `IsImage` na `true`, se poleg zapisa v bazi izbriše tudi pripadajoča slikovna datoteka.

Spreminjanje vrstnega reda korakov v projektu

Uporabnik lahko korakom v svojih projektih spreminja vrstni red prikaza. Na strani za urejanje dnevnika pritisne gumb z ikono seznama (na sliki 3.15 levo).

Ta na strani za urejanje dnevnika zažene način za spremjanje vrstnega reda korakov (na sliki 3.15 v sredini). Uporabnik v tem načinu pritisne na korak, ki ga želi prestaviti. Izbrani korak se obarva oranžno, nato pa se prekmika s pomočjo gumbov s puščicami (na sliki 3.15 desno). Nazaj v način za dodajanje se uporabnik vrne s pritiskom na gumb z ikono nalivnika.



Slika 3.15: Koda za ustvarjanje in pošiljanje zahtevka za dodajanje slikovnega koraka v aplikaciji.

Položaj koraka Note v projektu lahko razberemo iz atributa `Position`. Prvi korak ima `Position 0`, drugi `1`, tretji `2` itd.

```
Note {
    int Id;
    int Position;
    bool IsImage;
```

```
    string Title;
    string Text;
    int Hours;
    int Minutes;
    int Seconds;
    string Url;
    string ServerUrl;
}
```

Ob spremembi pozicije koraka v projektu ozadna koda aplikacije pošlje zahtevo PUT na URL „/projects/{pid}/{nid}/{positions}“.

Polje {pid} je enolični identifikator projekta, v katerem se nahaja korak. Polje {nid} je enolični identifikator koraka Note. Polje {positions} vsebuje število, ki pove, za koliko mest želimo korak prestaviti. To število je lahko pozitivno ali negativno celo število. Negativna vrednost {positions} prestavi korak proti začetku seznama, pozitivna pa proti koncu.

Implementacijo spreminjanja vrstnega reda korakov na strežniku prikazuje slika 3.16.

```

public async Task<Project> ChangeNotePositions(string userId, int projectId, int nid, int positions)
{
    var project = await GetByIdAsync(userId, projectId);
    var selectedNote = project.Notes.Where(n => n.Id == nid).FirstOrDefault();

    if (selectedNote == null)
    {
        return null;
    }

    int NoteIndex = selectedNote.Position;
    int LastIndex = NoteIndex + positions;

    selectedNote.Position = LastIndex;

    // korak premikamo proti začetku seznama
    if (positions < 0)
    [
        // zamakni korake, ki jih je spremenjeni "prehitel"
        foreach (Note n in project.Notes.Where(
            note => note.Position >= LastIndex &&
            note.Position < NoteIndex &&
            note.Id != selectedNote.Id
        ))
        {
            n.Position += 1;
        }
    ]
    // korak premikamo proti koncu seznama
    else
    {
        // zamakni korake, za katerimi je spremenjeni "zaostal"
        foreach (Note n in project.Notes.Where(
            note => note.Position <= LastIndex &&
            note.Position > NoteIndex &&
            note.Id != selectedNote.Id))
        {
            n.Position -= 1;
        }
    }

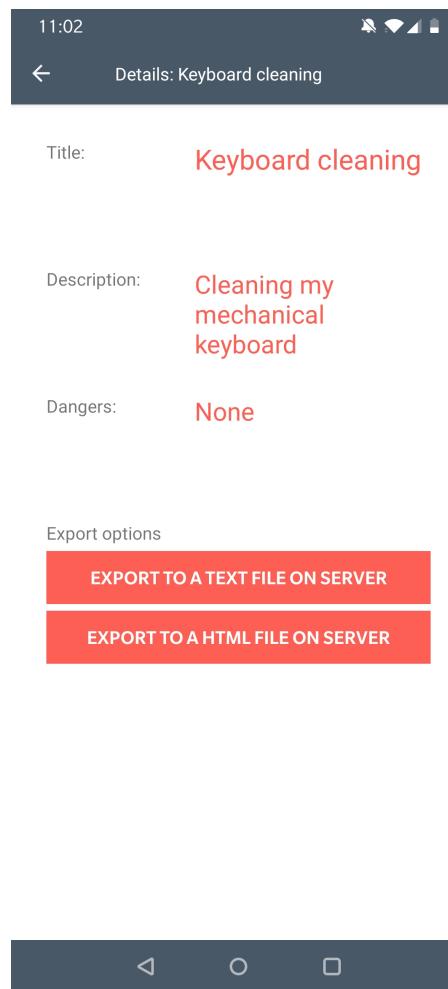
    await Save();
    return await GetByIdAsync(userId, projectId);
}

```

Slika 3.16: Koda za spremnjanje vrstnega reda korakov delavniškega dnevnika na strežniku.

3.4.6 Izvoz projektov

Sistem omogoča izvoz delavniških dnevnikov v druge računalniške formate. Izvoženi delavniški dnevnički lahko služijo kot varnostna kopija. Izvozimo jih lahko v besedilno datoteko ali v HTML dokument (slika 3.17).



Slika 3.17: Stran za izvoz projektov v mobilni aplikaciji.

Uporabnik projekt izvozi tako, da odpre stran s podrobnostmi dnevnika. To stran odpre s klikom na gumb z ikono zobnika na vrhu strani za prikaz projekta. Na strani s podrobnostmi dnevnika ima dva oranžna gumba (slika 3.17). S pritiskom na zgornji gumb izvozi projekt v besedilno datoteko, s pritiskom na spodnji gumb pa v HTML dokument.

V besedilno datoteko se projekt izvozi tako, da aplikacija pošlje GET zahtevo na URL „/projects/{id}/export/text“. Strežnik v besedilno datoteko vpiše naslov, opis projekta in možne nevarnosti pri delu. Nato ko-

rake uredi naraščajoče po vrednosti stolpca **Position** in jih enega za drugim zapiše v datoteko. Pri slikovnih korakih se zapiše besedilo in lokacija pripadajoče fotografije.

V HTML dokument se dnevnik izvozi tako, da aplikacija pošlje GET zahtevo na URL „/projects/{id}/export/html“. Pri izvozu v HTML dokument (slika 3.18) se naslov projekta zapiše kot HTML naslov H1, opis projekta kot naslov H2, koraki pa kot HTML odstavki. V HTML dokumentu lahko prikažemo poleg besedila slikovnih korakov tudi slike.

← → ⌂ ⌂ file:///C:/Users/Public/Documents/OpenReport/0_Keyboard_cleaning/20200811_21_02_34_Keyboard_cleaning.html

Keyboard cleaning

Description

Cleaning my mechanical keyboard

Dangers

None

1. Prepare the table and tools

Est. duration: 0 h 0 min 0 seconds

2. Remove keycaps

Use the tool, provided with the keyboard
Est. duration: 0 h 10 min 0 seconds

3. Put keycaps in soapy water

Leave in for 10 minutes
Est. duration: 0 h 10 min 0 seconds

4. Wipe keycaps until they are dry

Est. duration: 0 h 20 min 0 seconds

5. Place keycaps in their places



Est. duration: 0 h 5 min 0 seconds

Slika 3.18: Projekt, izvožen v HTML dokument.

Lokacija izvožene datoteke se nahaja v polju **FolderLocation** v razredu

Project. Prizeta lokacija, ki se nastavi ob ustvarjanju projekta, je „C:/users/\$USER/Public Documents/OpenReport“.

```
Project {  
    int Id;  
    string Title;  
    string Description;  
    string Dangers;  
    string FolderLocation; // lokacija pripadajočih datotek  
    IEnumerable<Note> Notes;  
    ...  
}
```

3.4.7 Uporaba kamere v Xamarin

Za uporabo kamere v Xamarin smo uporabili vtičnik Xam.Plugin.Media [14]. Vtičnik je licensiran pod MIT licenco.

Zanj smo se odločili, ker podpira zajem fotografij s privzeto aplikacijo za zajemanje fotografij. Izbiramo lahko tudi kvaliteto zajete fotografije.

Pred zajemom slike najprej preverimo, ali ima aplikacija na mobilni napravi zadostna dovoljenja. Imeti mora dovoljenje za dostop do shrambe, fotografij in kamere.

Fotografijo zajamemo z metodo `TakePictureAsync`. Zajeto fotografijo aplikacija hrani v objektu razreda `ImageSource`. Fotografijo v tej obliki se lahko prikaže v XAML z grafičnim elementom `Image`. Lahko jo zakodiramo v znakovni niz in pošljemo na strežnik.

3.5 Uvedba glasovnega pomočnika

Po implementaciji strežnika in mobilne aplikacije smo se lotili uvajanja glasovnega pomočnika v sistem OpenReport.

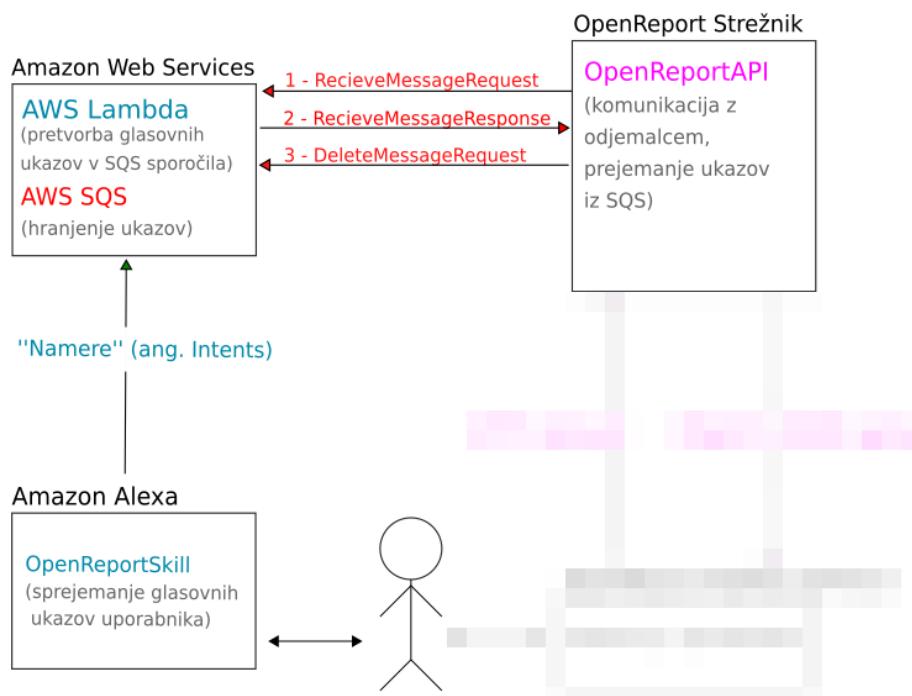
3.5.1 Funkcionalne zahteve Alexa veščine

Alexina veščina je program, ki glasovnemu pomočniku Amazon Alexa doda funkcionalnosti. Alexo in zasnovano veščino smo uporabili kot dodaten uporabniški vmesnik za upravljanje z mobilno aplikacijo.

Z uvedbo Alexe smo želeli doseči:

- narekovnaje besedilnega koraka,
- prostoročno odpiranje obrazca za dodajanje korakov v odprt projekt in
- prostoročno odpiranje kamere in dodajanje slikovnega koraka v projekt.

3.5.2 Komunikacija strežnika z Amazon Alexo



Slika 3.19: Komunikacija med strežnikom in glasovnim pomočnikom preko SQS.

Strežnik z glasovnim pomočnikom Amazon Alexa komunicira (slika 3.19) preko storitve Simple Queue Service (SQS). Amazon Alexa rezultate obdelanih glasovnih ukazov odloži v SQS vrsto, strežnik pa jih iz te vrste prebere.

SQS vrsta FIFO je sklad besedilnih sporočil. Nanjo lahko pošiljamo nova sporočila ter beremo in brišemo obstoječa sporočila.

Glasovni pomočnik od uporabnika prejme izgovorjen glasovni ukaz. Na podlagi tega ukaza glasovni pomočnik na SQS vrsto odloži tri različna besedilna sporočila:

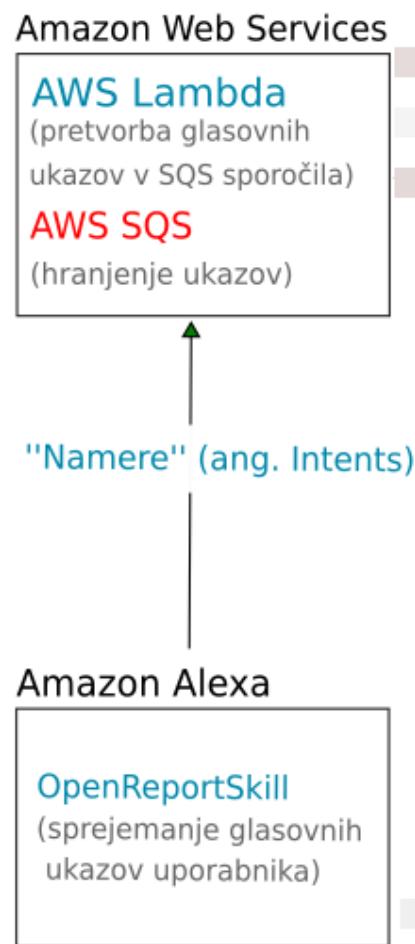
- Sporočilo „**addnote**“ sporoči strežniku, naj na odjemalcu odpre obrazec za dodajanje besedilnega koraka.
- Sporočilo „**addimage**“ sporoči strežniku, naj na odjemalcu odpre kamero za dodajanje slikovnega koraka.
- Sporočilo „**{narekovano besedilo}**“ sporoči strežniku, naj vsebino tega sporočila doda odprtemu projektu kot besedilni korak.

3.5.3 Implementacija Alexa veščine

Poziv, ki ga mora uporabnik izreči za zagon implementirane veščine se glasi „*make a report note*“.

Vsak nadaljnji ukaz, ki ga uporabnik izreče, preden se veščina preneha izvajati, se primerja s frazami namer. Namere se uporabljamjo za zagon funkcij, ki smo jih implementirali v krajišču (slika 3.20). V naši veščini smo definirali tri glavne namere:

- `TakeNoteIntent`, ki omogoča narekovanje besedila korakov,
- `OpenTextNoteFormIntent`, ki omogoča odprianje obrazca za dodajanje koraka in
- `OpenImageNoteFormIntent`, ki omogoča zajem fotografije v aplikaciji.

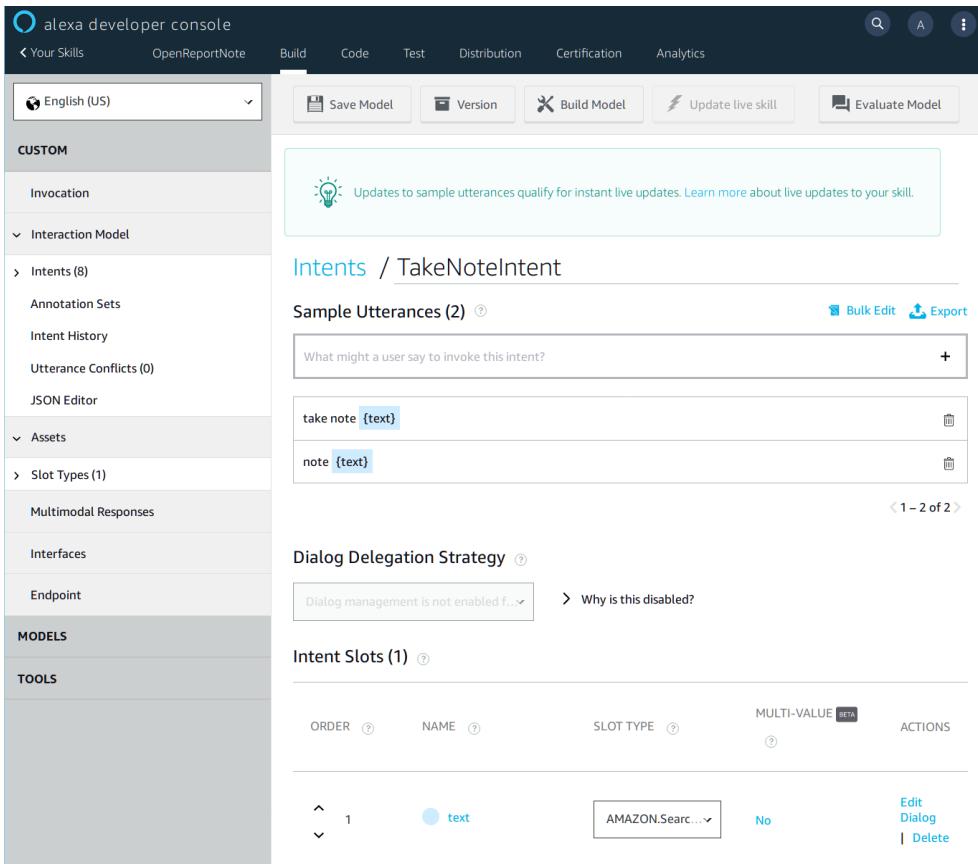


Slika 3.20: Načrt komunikacije Alexe, AWS Lambda in AWS SQS.

Narekovanje koraka

Uporabnik korak delavnškega dnevnika narekuje tako, da Alexi izgovori poziv „make a report note“ in nadaljuje s „take note {besedilo}“. Polje **{besedilo}** je spremenljivka, v katero veščina hrani razpoznano besedilno vsebino koraka. V ozadni kodi veščine se zažene namera **TakeNoteIntent** (slika 3.21).

Prikaz uporabe: uporabnik po pozivu izreče „note unscrew the backplate“, torej vrednost spremenljivke **{besedilo}** postane „unscrew the backplate“.



Slika 3.21: Frazi TakeNoteIntent v Alexa Developer nadzorni plošči.

Namera `TakeNoteIntent` ustvari novo SQS sporočilo, vanj vstavi vrednost spremenljivke `{besedilo}` in ga pošlje na SQS vrsto (implementacija na sliki 3.24). To SQS sporočilo prebere strežnik in ga kot korak vstavi v odprt projekt. Večina vrne uporabniku odgovor „Noted!“ in se preneha izvajati.

Odpiranje obrazca za dodajanje besedilnega koraka

Uporabnik v mobilni aplikaciji prostoročno odpre obrazec za dodajanje besedilnega koraka tako, da Alexi izgovori poziv naše veščine in nadaljuje s „create a text note“ ali „add a text note“ (slika 3.22).

V ozadni kodi veščine se zažene namera `OpenTextNoteFormIntent`, ki ustvari novo SQS sporočilo, vanj vstavi besedilo „`addnote`“ in ga pošlje v SQS vrsto. To SQS sporočilo prebere strežnik in mobilni aplikaciji sporoči, naj odpre obrazec za dodajanje novega besedilnega koraka. Veščina zatem vrne uporabniku odgovor „Opening the form!“ in se preneha izvajati.

Intents / OpenTextNoteFormIntent

Sample Utterances (2) ?

What might a user say to invoke this intent?

create a text note

add a text note

Slika 3.22: Frazi `OpenTextNoteFormIntent` v Alexa Developer nadzorni plošči.

Odpiranje kamere in dodajanje slikovnega koraka

Uporabnik kamero prostoročno odpre tako, da Alexi izgovori poziv naše veščine in nadaljuje s „`take a picture`“ (slika 3.23).

V ozadni kodi veščine, se zažene namera `OpenImageNoteFormIntent` (slika 3.23). `OpenImageNoteFormIntent` ustvari novo SQS sporočilo, vanj vstavi besedilo „`addimage`“ in ga pošlje v SQS vrsto. To SQS sporočilo prebere strežnik in mobilni aplikaciji sporoči, naj odpre kamero. Veščina zatem vrne uporabniku odgovor „`Launching camera!`“ in se preneha izvajati.

Intents / OpenImageNoteFormIntent

Sample Utterances (1) ①

What might a user say to invoke this intent?

take a picture

Slika 3.23: Fraza OpenImageNoteFormIntent v Alexa Developer nadzorni plošči.

```
if (input.GetRequestType() == typeof(LaunchRequest))
{
    string speech = "What now?";
    Reprompt rp = new Reprompt("Say \"Take a picture\", \"Create a text note\" or \"Take note\" and cite your note");
    return ResponseBuilder.Ask(speech, rp, session);
}
else if (input.GetRequestType() == typeof(SessionEndedRequest))
{
    return ResponseBuilder.Tell("Stopping.");
}
else if (input.GetRequestType() == typeof(IntentRequest))
{
    var IntentRequest = (IntentRequest)input.Request;
    switch (IntentRequest.Intent.Name)
    {
        case "AMAZON.CancelIntent":
        case "AMAZON.StopIntent":
            return ResponseBuilder.Tell("Stopping.");
        case "AMAZON.HelpIntent":
            return ResponseBuilder.Tell("Start by saying \"Note\". Then start talking!");
        case "TakeNoteIntent":
            {
                string userString = IntentRequest.Intent.Slots["text"].Value;
                try
                {
                    smr = new SendMessageRequest
                    {
                        QueueUrl = "https://sqs.us-west-2.amazonaws.com/993067550608/NoteQueue.fifo",
                        MessageBody = userString.ToString(),
                        MessageGroupId = "1",
                        MessageDuplicationId = Guid.NewGuid().ToString(),
                    };
                    smrResponse = await sqsClient.SendMessageAsync(smr);
                    return ResponseBuilder.Tell("Noted!");
                }
                catch (Exception ex)
                {
                    return ResponseBuilder.Tell(ex.Message);
                }
            }
    }
}
```

Slika 3.24: Odsek ozadne kode Alexine veščine s prikazom ustvarjanja SQS sporočila.

Prenašanje sporočil med strežnikom in SQS

Komunikacija OpenReport strežnika z SQS je prikazana z rdečo barvo na sliki 3.25.



Slika 3.25: Komunikacija strežnika z SQS.

Na strežniku teče storitev za komunikacijo z glasovnim pomočnikom. Ta storitev vsakih pet sekund na SQS pošlje zahtevek RecieveMessageRequest.

```

RecieveMessageRequest {
    AttributeName
    MaxNumberOfMessages
    QueueUrl
    WaitTimeSeconds
    ...
}
  
```

AWS SQS kot odgovor vrne objekt tipa RecieveMessageResponse.

```

RecieveMessageResponse {
    IEnumerable<Message> Messages;
    ...
}
  
```

V tem odgovoru se nahaja seznam sporočil, ki čakajo na sprejem iz SQS vrste. V kolikor je v odgovoru vsaj eno sporočilo, pregledamo telo vseh sporočil.

Ko strežnik sporočila sprejme, jih mora izbrisati iz SQS vrste. Če jih sproti ne izbriše, iz FIFO vrste ne more brati najnovejših sporočil, ampak le 5 najstarejših.

Sporočila izbriše iz SQS vrste z zahtevkom `DeleteMessageRequest`. Ta zahtevek hrani referenco na sporočilo, ki ga želimo izbrisati iz SQS vrste. AWS SQS storitev v odgovor vrne `DeleteMessageResponse`, a v tej diplomski nalogi tega odgovora ne uporabimo več.

Zahtevki, ki jih strežnik prejme od glasovnega pomočnika, se odražajo na zadnjem projektu, ki ga uporabnik z glasovnim pomočnikom odpre.

- Zahtevek „`addnote`“ nastavi zastavico „`NoteRequest`“ na `true`,
- zahtevek „`addimage`“ nastavi zastavico „`ImageRequest`“ na `true`,
- zahtevek „`{narekovano besedilo}`“ nastavi zastavico „`AlexaNoteRequest`“ na `true` in prejeto besedilo doda projektu kot korak.

```
Project {
    int Id;
    string Title;
    string Description;
    string Dangers;
    string FolderLocation;
    IEnumerable<Note> Notes;
    bool NoteRequest;
    bool ImageRequest;
    bool AlexaNoteRequest;
}
```

Če je vsebina prejetega sporočila enaka „`addnote`“, storitev preveri, kateri uporabnik ima trenutno nase vezanega glasovnega pomočnika. Nato v

projektu, ki ga je nazadnje odprl, nastavi vrednost zastavice `NoteRequest` na `true`.

Če je vsebina sporočila enaka „`addimage`“, storitev preveri, kateri uporabnik ima trenutno nase vezanega glasovnega pomočnika. V projektu, ki ga je uporabnik nazadnje odprl, nastavi vrednost zastavice `ImageRequest` na `true`.

Če vsebina prejetega sporočila ni enaka „`addimage`“ ali „`addnote`“, je prejeto sporočilo dobesedno narekovani korak. Storitev preveri, kateri uporabnik ima nase vezanega glasovnega pomočnika. V projektu, ki ga je ta uporabnik nazadnje odprl, nastavi vrednost zastavice `AlexaNoteRequest` na `true`. V ta projekt se vstavi korak, ki ima naslov „Voice note“, vsebina pa je telo prejetega sporočila.

Utemeljitev uporabe SQS

Prenos sporočil med Amazon Alexo in strežnikom smo realizirali z uporabo storitve AWS SQS. Pri uporabi SQS je prenos podatkov resda manj učinkovit kot pri direktni povezavi preko spletne vtičnice (*ang. Web Socket*).

Glavni faktor, zaradi katerega smo se odločili za uporabo AWS SQS, je, da z dodatno plastjo za hranjenje sporočil dosežemo enostavno zamenljivost glasovnega pomočnika.

Pri morebitni menjavi glasovnega pomočnika strežniškega programa ne bi spremajali. Potrebno bi bilo le implementirati program za novega glasovnega pomočnika, da na AWS SQS odlaga sporočila v strežniku razumljivem formatu („`addnote`“, „`addimage`“).

3.5.4 Ovire pri izdelavi Alexa veščine

Pri implementaciji veščine smo naleteli na veliko težav.

Prva je bila jezikovne narave, saj Alexa ne podpira prepoznavanja slovenskega jezika, zato smo se odločili za uporabo angleškega jezika.

Naslednja težava je bila definicija pozivne fraze. Fraze, kot so „take a

note“ ali „take a picture“, so že rezervirane v sklopu Alexinih privzetih funkcionalnosti, kar pomeni, da jih za našo večino ne moremo uporabiti.

Izogibati smo se morali tudi frazam, ki so bile rezerviranim frazam podobne (npr. „open report note“ itd.). Tudi take fraze so se velikokrat razpoznale kot rezervirane fraze.

Težavo je predstavljala izbira poziva, ki je bil hkrati kratek, intuitiven in se ni napačno interpretiral v poziv privzetih funkcionalnosti. Poziv „make a report note“ je bil najboljši kompromis, ki smo ga našli.

Težavo je predstavljalo tudi razpoznavanje dolgih stavkov, še posebej če se le-ti niso ujemali z definiranimi frazami namer. To je zelo omejilo možnost narekovanja besedila korakov s pomočjo Amazon Alexe.

Naslednjo težavo je predstavljal čas, ki ga je Alexa porabila za interpretacijo definiranih glasovnih ukazov. Za procesiranje enega glasovnega ukaza je porabila od 3 do 6 sekund.

3.6 Dodelave mobilne aplikacije

Po implementaciji Alexa večine smo v mobilni aplikaciji implementirali storitev za upravljanje z glasovnim pomočnikom. Ta storitev uporabniku omogoča, da lahko zahteva Alexa za delo s svojimi projekti in jo po uporabi sprosti.

3.6.1 Zahtevanje in sprostitev glasovnega pomočnika

OpenReport strežnik podpira delo z enim glasovnim pomočnikom. Pomočnika lahko uporabnik zahteva za lastno uporabo in ga po uporabi sprosti.

Ko se zahtevek za uporabo pomočnika potrdi, ga lahko uporabnik uporablja pri sestavljanju delavnškega dnevnika. Uporabnik lahko glasovnega pomočnika sprosti, kar omogoči drugim uporabnikom, da ga lahko zahtevajo za svoje delo.

Ali ima določen uporabnik nase vezanega pomočnika, vidimo po vrednosti zastavice `UsesVoiceAssistant` v njegovem objektu razreda `User`.

Ko uporabnik odpre katerega od svojih projektov, se enolični identifikator tega projekta nastavi v polje `LastOpenedProjectId` v njegovem objektu razreda `User`. Zahtevki, ki jih strežniku pošlje Alexa preko SQS vrste, se navezujejo na projekt s tem identifikatorjem.

```
User : IdentityUser {  
    string Id;  
    string Email;  
    string Password;  
    IEnumerable<Project> Projects;  
    ...  
    int LastUsedProjectId;  
    bool UsesVoiceAssistant;  
}
```

Uporabnik je pomočnika uspešno zahteval zase, če je `UsesVoiceAssistant` nastavljen na `True`. V mobilni aplikaciji zahtevanje in sprostitev pomočnika nadziramo na Dashboard strani s pritiskom na ikono mikrofona (slika 3.26).

V aplikaciji za zahtevanje in sprostitev pomočnika skrbi razred `VoiceAssistantService.cs`. Primerek tega razreda se ustvari ob začetku izvajanja aplikacije, nato pa počaka, da avtentikacijska storitev mobilne aplikacije uspešno opravi prijavo uporabnika.

Če ima uporabnik vrednost polja `UsesVoiceAssistant` nastavljeno na `true`, ima glasovnega pomočnika vezanega nase in ga lahko uporablja.

Če ima uporabnik vrednost polja `UsesVoiceAssistant` nastavljeno na `false`, mora uporabo glasovnega pomočnika zahtevati. To uporabnik doseže tako, da pritisne na gumb z ikono prečrtanega mikrofona na Dashboard strani (na sliki 3.26 desno). Ozadna koda aplikacije pošlje avtoriziran PUT zahtevek na URL „/users/requestva“.

Če nihče od drugih prijavljenih uporabnikov nima polja `UsesVoiceAssistant` nastavljenega na `true`, se njegova zahteva odobri. V njegovem objektu razreda `User` se nastavi vrednost `UsesVoiceAssistant` na `true`.

Če ima kateri od uporabnikov to polje nastavljeno na `true`, zahteva ne bo odobrena. V tem primeru mora uporabnik, ki zahteva pomočnika počakati, da uporabnik, ki pomočnika uporablja, le tega sprosti. Ko si pomočnika ne lasti nihče drug, ga lahko uporabnik zahteva zase.

Uporabnik pomočnika sprosti, ko na strani Dashboard v aplikaciji pritisne gumb z ikono mikrofona. Ozadna koda aplikacije pošlje PUT zahtevek na URL „/users/releaseva“. Na strežniku se nato vrednost polja `UsesVoiceAssistant` za pošiljatelja zahteve nastavi na `false`.



Slika 3.26: Gumb v obeh stanjih.

Poglavlje 4

Ovrednotenje funkcionalnosti

Sistem OpenReport smo po končani implementaciji testirali. Mobilna aplikacija in strežnik sta izpolnila pričakovanja.

Z mobilno aplikacijo se je bilo možno povezati na strežnik. Registracija in prijava v sistem preko mobilne aplikacije nista povzročala težav. Brez težav je delovalo tudi zahtevanje in sprostitev glasovnega pomočnika Amazon Alexe na strani Dashboard.

Delavniške dnevниke je bilo brez težav mogoče dodajati, odpirati in bri-sati.

Dodajanje besedilnih korakov v delavniški dnevnik smo ocenili kot smiselno za manjše opombe. Možnost dodajanja slikovnih korakov v delavniški dnevnik se je izkazala za dobro pri pomembnejših opornih točkah.

Naknadno spremenjanje naslovov, trajanja in besedila korakov ni povzročalo problemov. Urejanje vrstnega reda korakov je delovalo brez težav. Tudi brisanje korakov delavniškega dnevnika ni povzročalo težav.

Možnost izvoza delavniškega dnevnika v druge formate je delovala brez nevšečnosti. Ta funkcija lahko olajša prenos podatkov v druge računalniške programe.

Alexino veščino smo testirali tako, da smo vsako frazo namere izgovorili 20-krat in beležili, kolikokrat se je razpoznaла pravilno. Pri dobesednem narekovaju koraka smo testiranje opravili dvakrat. Prvič smo testirali s

stavkom „*this is a voice note*“, drugič s stavkom „*removing the backplate takes about three minutes*“, tretjič pa s stavkom „*prepare the workspace*“.

- Alexa je ukaz „*take a picture*“ pravilno razpoznaла v 18 od 20 testnih izgovorjav.
- Ukaz „*create a text note*“ je pravilno razpoznaла v 17 od 20 testnih izgovorjav.
- Ukaz „*take note this is a voice note*“ je pravilno razpoznała v 6 od 20 testnih izgovorjav.
- Ukaza „*take note removing the backplate takes about three minutes*“ ni pravilno razpoznała v nobeni od 20 testnih izgovorjav.
- Ukaz „*take note prepare the workspace*“ je pravilno razpoznała v 11 od 20 testnih izgovorjav.

Narekovanje besedila glasovnemu pomočniku ni v celoti izpolnilo pričakovanj. Alexa pri narekovaju besedila v večini primerov ni pravilno razpoznała namere. Rezultati testiranja razpoznavanja glasovnih ukazov so pokazali, da v našem primeru Alexa veliko bolje razpoznavata namere, ki se z definirano frazo namere v celoti ujemajo kot tiste, ki nimajo velike stopnje ujemanja.

Dodajanje slikovnih in besedilnih korakov s pomočjo glasovnega ukaza se je izkazalo za delno uspešno. Alexa je v teh primerih ukaze razpoznavala z zadovoljivo natančnostjo. Za uporabno se je izkazalo predvsem, ko mobilnega telefona med delom nismo imeli na dosegu roke. Vseeno pa je predstavljal težavo čas, ki je bil potreben, za izgovorjavo poziva, fraz namer in čakanje na obdelavo zahtev. To je trajalo v povprečju 18 sekund, kar smo ocenili kot prepočasno za učinkovito uporabo.

Odklepanje mobilnega telefona, ki smo ga imeli v dosegu roke ter odpiranje obrazca za dodajanje korakov, je trajalo v povprečju 4 sekunde. Odklepanje mobilnega telefona, ki je bil oddaljen tri metre ter odpiranje obrazca za dodajanje korakov, je trajalo v povprečju 8 sekund.

Težavo je predstavljalo tudi, če uporabnik ni znal angleškega jezika.

Naposled smo ugotavili, da je za zasnovno sistema, ki vključuje Amazon Alexa, treba skrbno upoštevati Alexine omejitve. V našem primeru smo dobro prepoznavanje ukazov dosegli pri ukazih brez spremenljivih členov. Sklepamo lahko, da bi se sprejemljivo obnesla v sistemih, kjer so glasovni ukazi statično definirani.

Kljub nekaterim pomanjkljivostim pa smo razvili funkcionalen sistem, ki lahko služi kot dobra osnova za nadaljnji razvoj. Sistem smo namreč lahko uporabljali tudi brez glasovnega pomočnika za učinkovito sestavljanje delavninskih dnevnikov.

Poglavlje 5

Zaključek

Sistem, razvit v sklopu diplomske naloge, zajema funkcionalosti, ki uporabniku omogočajo sestavljanje delavniških dnevnikov. Sestavljajo ga strežnik, mobilna aplikacija in glasovni pomočnik Amazon Alexa. Uporabnik lahko v delavniške dnevниke dodaja preproste besedilne korake. Z uporabo mobilne aplikacije lahko zajame fotografije in jih v delavniški dnevnik vstavi kot slikovno gradivo.

Ta sistem je služil kot primer sistema, ki ga želimo izboljšati z uporabo glasovnega pomočnika. Naš namen je bil poenostaviti delo s sistemom z uvedbo glasovnega pomočnika Amazon Alexe. Implementirali smo odpiranje kamere in obrazca za dodajanje korakov z glasovnim pomočnikom. Kljub uspešni implementaciji je bil čas, potreben za izgovorjavo in obdelavo glasovnih ukazov, predolg, da bi poskus izboljšave bil popolnoma uspešen. Za neuspešno se je izkazalo glasovno narekovanje vsebine koraka delavniškega dnevnika. Z Amazon Alexo ni bilo mogoče razpozнатi daljših, spremenljivih stavkov.

V možne izboljšave sistema smo na prvo mesto uvrstili preizkus drugih glasovnih pomočnikov, npr. Google Assistant [6]. Druga smiselna izboljšava je izvoz delavniških dnevnikov v več različnih standardnih formatov, kot so npr. XSL ali ODS.

Mobilna aplikacija in strežniški program sta zadovoljila pričakovanja in

lahko že v trenutnem stanju služita kot referenca za primerjavo učinkovitosti Amazon Alexe in drugih glasovnih pomočnikov.

Kljub nekaterim težavam z uporabo glasovnega pomočnika je očitno, da je tehnologija glasovnih pomočnikov zelo primeren način za nadzor sistemov na daljavo. Poleg tega ponuja veliko možnosti za slepe, slabovidne in gibalno omejene posameznike, ki imajo težave pri interakciji z računalnikom.

Literatura

- [1] *Amazon Alexa*. Dosegljivo: <https://www.amazon.com/b?ie=UTF8&node=17934671011>. [Dostopano: 12. 8. 2020].
- [2] Jonas Austerjost, Marc Porr, Noah Riedel, Dominik Geier, Thomas Becker, Thomas Scheper, Daniel Marquard, Patrick Lindner in Sascha Beutel. „Introducing a virtual assistant to the lab: A voice user interface for the intuitive control of laboratory instruments“. V: *SLAS TECHNOLOGY: Translating Life Sciences Innovation* 23.5 (2018), str. 476–482.
- [3] *Funkcionalnosti .NET ogrodja*. Dosegljivo: <https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet>. [Dostopano: 18. 8. 2020].
- [4] *Funkcionalnosti AWS Lambda*. Dosegljivo: <https://aws.amazon.com/lambda/features/>. [Dostopano: 12. 8. 2020].
- [5] *Funkcionalnosti AWS SQS*. Dosegljivo: <https://aws.amazon.com/sqs/features/>. [Dostopano: 12. 8. 2020].
- [6] *Google Assistant*. Dosegljivo: <https://assistant.google.com/>. [Dostopano: 18. 8. 2020].
- [7] *LibreOffice Writer*. Dosegljivo: <https://www.libreoffice.org/discover/writer/>. [Dostopano: 12. 8. 2020].
- [8] *Navodila za pisanje delavnškega dnevnika ŠCLJ*. Dosegljivo: http://www.lesarska.sclj.si/images/Pravilniki/NOZ_PRA.doc. [Dostopano: 28. 8. 2020].

- [9] *Podrobnosti SQS FIFO vrste.* Dosegljivo: <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html>. [Dostopano: 30. 8. 2020].
- [10] *Shema MVVM pristopa.* Dosegljivo: <https://docs.devexpress.com/WPF/images/winforms-mvvm-common-mvvm-scheme118014.png>. [Dostopano: 18. 8. 2020].
- [11] *Slika arhitekture Alexine veščine.* Dosegljivo: https://m.media-amazon.com/images/G/01/DeveloperBlogs/AmazonDeveloperBlogs/legacy/cloudformation_image12._CB520203781_.png. [Dostopano: 28. 8. 2020].
- [12] Gospodarska Zbornica Slovenije. *Delavniški dnevnik kot izobraževalni pripomoček pri vajenju.* Dosegljivo: https://www.gzs.si/Portals/203/Vsebine/novice-priponke/DNEVNIKzavajenistvo_2019_navodilazzgledom.pdf. [Dostopano: 3. 9. 2020].
- [13] *Uvod v Alexine veščine.* Dosegljivo: <https://developer.amazon.com/en-US/docs/alexa/ask-overviews/build-skills-with-the-alexa-skills-kit.html>. [Dostopano: 5. 9. 2020].
- [14] *Xam.Plugin.Media.* Dosegljivo: <https://github.com/jamesmontemagno/MediaPlugin>. [Dostopano: 12. 8. 2020].
- [15] *Xamarin.* Dosegljivo: <https://dotnet.microsoft.com/apps/xamarin>. [Dostopano: 12. 8. 2020].
- [16] *Žetoni JWT kot sredstvo za avtorizacijo.* Dosegljivo: <https://jwt.io/introduction/>. [Dostopano: 28. 8. 2020].